TUGAS AKHIR – KI141502

PEMBUATAN KAKAS BANTU PENGUKURAN KUALITAS PERANGKAT LUNAK PADA KODE PEMROGRAMAN JAVA.

Wati Margaretha Marpaung
NRP 5111100 200

Dosen Pembimbing
Dr. Ir. Siti Rochimah, M.T.
Daniel Oranova Siahaan, S.Kom., M.Sc., P.D.Eng.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2015

FINAL PROJECT – KI141502

DEVELOPMENT OF TOOLS FOR QUALITY MEASUREMENTS FOR JAVA PROGRAMMING CODE

Wati Margaretha Marpaung
NRP 5111100 200

Advisor
Dr. Ir. Siti Rochimah, M.T.
Daniel Oranova Siahaan, S.Kom., M.Sc., P.D.Eng.

DEPARTMENT OF INFORMATICS
Faculty of Information Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2015

LEMBAR PENGESAHAN

Pembuatan Kakas Bantu Prngukuran Kualitas Perangkat Lunak pada Kode Pemrograman Java

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Rekayasa Perangkat Lunak
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh :
WATI MARGARETHA MARPAUNG
NRP : 5111 100 200

Disetujui oleh Dosen Pembimbing tugas akhir :

Dr. Ir. SITI ROCHIMAH, M.T. .....
NIP: 19681002199403 2 001 (pembimbing 1)

DANIEL ORANOVA SIAHAAN, ...
S.Kom., M.Sc., P.D.Eng (pembimbing 2)
NIP: 19741123200604 1 001

SURABAYA
JANUARI 2015

Pembuatan Kakas Bantu Pengukuran Kualitas Perangkat Lunak pada Kode Pemrograman Java

Nama Mahasiswa : Wati Margaretha Marpaung
NRP : 5111100200
Jurusan : Teknik Informatika FTIf-ITS
Dosen Pembimbing 1: Dr. Ir. Siti Rochimah, M.T.
Dosen Pembimbing 2: Daniel O. Siahaan, S.Kom., M.Sc., P.D.Eng

ABSTRAK

Pengujian perangkat lunak adalah kegiatan yang ditujukan untuk mengevaluasi atribut. Hal ini juga berarti memastikan kemampuan program tersebut memenuhi hasil yang dicari atau suatu investigasi yang dilakukan untuk mendapatkan informasi mengenai kualitas yang sedang diuji (under test). Pengujian perangkat lunak secara objektif dan independen bermanfaat dalam operasional bisnis utnuk memahami tingkat resiko pada implementasinya.

Pengembangan perangkat lunak dan desain berorientasi objek menjadi sangat populer dalam pengembangan perangkat lunak saat ini. Pengembangan berorientsi objek tidak hanya membutuhkan pendekatan yang berbeda untuk merancang dan menerapkannya. Metrik-metrik yang terdapat dalam jenis pemrograman ini antara lain adalah Kompleksitas Siklomatik (Cyclomatic Complexity), Ukuran Kode (Size) meliputi perhitungan fisik baris kode, Persentasi Komentar, Weighted Method Class (WMC), Response For Class (RFC), Number of Subunits, Coupling Between Object Classes (CBO), Depth of Inheritance Tree (DIT), dan Number of Children (NOC).

Bagian-bagian program ini berada dalam sistem dan dapat diukur keberadaanya. Guna mendeteksi adanya metrik tersebut pada progam diperlukan teknik untuk melakukan code analysis.

Kakas ini mendeteksi metrik-metrik pada kode program Java yang dibangun menggunakan framework Java. Kode program sebagai inputan dalam bentuk .java maupun .txt diurai kemudian dimanfaatkan untuk melakukan analisis terhadap kode program. Proses analisis tersebut kemudian menghasilkan range serta grafik kualitas perangkat lunak yang diuji. Kakas telah diuji coba pada 3 aplikasi java dengan hasil

rata-rata nilai yang sesuai dengan paper menunjukkan bahwa sistem dapat mendeteksi seluruh metrik yang akan diuji dengan baik.

Kata kunci**: **Pengujian Perangkat Lunak, Metrik ukur Perangkat Lunak, Java, Cyclomatic Complexity, Size, Comment Percentage, Weighted Method Class, Response For Class, Number of Subunits, Coupling between Objects, Depth of Inheritance Tree, Number of Children .

Development of Tools for Quality Measurements for Java Programming Code

Student Name	: Wati Margaretha Marpaung
Student ID	: 5111100200
Major	: Teknik Informatika FTIf-ITS
Advisor 1	: Dr. Ir. Siti Rochimah, M.T.
Advisor 2	: Daniel O. Siahaan, S.Kom., M.Sc., P.D.Eng

ABSTRACT

Software testing is an activity with the purpose to evaluate attributes or a program's capability and make sure that the software is fulfiillng the requirement, or an investigation that made to get the information about the quality of product or services that being tested. An objective and independent testing will be usefull for operational business to understand the risk that might happen at the implementation time.

Object oriented software development and design became very popular as a metodhology to develop a software nowadays. Object oriented development not only need a different approach when we design or implement it, but also when we try to test it. Some metrics that we can use for this methodology are Cyclomatic Comlexity, Size which contains all the calculation of the physical line of code, Comment percentage, Weighted Method Class(WMC), Response For Class (RFC), Number of Subunits (NOS), Coupling Between Object (CBO), Depth of Inheritance Tree (DIT), and Number of Children (NOC).

This metrics are parts of program that lies within the system and are countable. To detect and calculate the value of the metrics inside a program, we need a technic to do the code analysis.

This tool will detect all those metrics within a java code and it built using java framework. The source code as the input in .java or .txt format will be parsed and then will be used to do the analysis of the program. The analysis process will then produce range and barchart to show the quality of the code. This tools have been tested on 3 java application which average result is the same with papers that being used as reference which show that the tools can detect all the metrics and giving the right results.

***Keyword**: Software Testing, Software Metrics Measurement, Java, Cyclomatic Complexity, Size, Non-Comment, Comment Percentage, Weighted Method Class, Response For Class, Number of Subunits, Coupling between Objects, Depth of Inheritance Tree, Number of Children .*

KATA PENGANTAR

Puji syukur kepada Allah Yang Maha Esa atas segala karunia dan rahmat-Nya penulis dapat menyelesaikan tugas akhir yang berjudul :

"Pembuatan Kakas Bantu Pengukuran Kualita Perangkat Lunak Pada Kode Pemrograman Java"

Harapan dari penulis semoga apa yang tertulis di dalam buku tugas akhir ini dapat bermanfaat bagi pengembangan ilmu pengetahuan saat ini, serta dapat memberikan kontribusi yang nyata.

Dalam pelaksanaan dan pembuatan tugas akhir ini tentunya sangat banyak bantuan yang penulis terima dari berbagai pihak, tanpa mengurangi rasa hormat penulis ingin mengucapkan terima kasih sebesar-besarnya kepada:

1. Tuhan Yesus Kristus yang memberikan berkatnya kepada saya untuk menyelesaikan tugas akhir ini dan menyelesaikan pendidikan saya di Teknik Informatika ITS selama ini. Bunda Maria, sebagai teladan saya untuk bersabar dalam segala proses yang saya lalui setiap hari.
2. Bapak, Mamak, Bang Bangun, Bang Kardo, Theresia dan keluarga yang selalu memberikan dukungan penuh untuk menyelesaikan tugas akhir ini.
3. Bu Siti Rochimah dan Bapak Daniel O. Siahaan selaku dosen pembimbing yang telah bersedia meluangkan waktu untuk memberikan petunjuk selama proses pengerjaan tugas akhir ini.
4. Mr. Edison Marpaung, S.Kom. sebagai partner utama saya yang mau membimbing dan mengarahkan saya mulai dari menginjakkan kaki di kampus perjuangan sampai akhir.
5. Bapak Imam Kuswardayan sebagai dosen wali serta seluruh Bapak, Ibu dosen Jurusan Teknik Informatika ITS yang telah banyak memberikan ilmu dan bimbingan selama 4 tahun di Teknik Informatika ITS kepada penulis.

6. Seluruh staff dan karyawan FTIF ITS yang banyak memberikan kelancaran administrasi akademik kepada penulis.
7. Teman-teman penghuni Lab IGS dan teman-teman lain yang telah memberikan banyak dukungan dan semangat kepada penulis.
8. Santi, Yoko dan teman-teman angkatan 2011 jurusan Teknik Informatika ITS yang telah menemani perjuangan selama 4 tahun ini atas saran, masukan, dan dukungan terhadap pengerjaan tugas akhir ini.
9. Alumni Budi Mulia Siantar Surabaya (ABISS), KMK ITS, Beswand 29 terkhususnya Budjangan Surabaya, HMTC terkhususnya C1B, dan MBP ITS.
10. Serta seluruh pihak yang tidak dapat saya sebutkan satu-persatu yang telah banyak membantu penulis dalam penyusunan tugas akhir ini.

Penulis telah berusaha sebaik-baiknya dalam menyusun tugas akhir ini, namun penulis mohon maaf apabila terdapat kekurangan, kesalahan maupun kelalaian yang telah penulis lakukan. Kritik dan saran yang membangun dapat disampaikan sebagai bahan perbaikan selanjutnya.

Surabaya, Mei 2015

Wati Margaretha Marpaung

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR KODE SUMBER

BAB I
PENDAHULUAN

Pada bab ini akan dipaparkan mengenai garis besar tugas akhir yang meliputi latar belakang, tujuan, rumusan dan batasan permasalahan, metodologi pembuatan tugas akhir, dan sistematika penulisan.

1.1. Latar Belakang

Pengembangan perangkat lunak dengan menggunakan konsep orientasi objek memudahkan desainer dalam membangun perangkat lunak. Konsep orientasi objek juga telah memacu desainer dalam mengembangkan perangkat menggunakan berbagai teknik dan pola-pola implementasi. Salah satu contoh penggunaan konsep orientasi objek adalah pada kode pemrograman Java. Beragamnya pola implementasi orientasi objek juga menimbulkan perbedaan pendapat dalam melihat kualitas sebuah perangkat lunak.

Mengukur kualitas perangkat lunak memang bukan pekerjaan mudah. Ketika seseorang memberi nilai sangat baik terhadap sebuah perangkat lunak, orang lain belum tentu mengatakan hal yang sama. Sudut pandang seseorang tersebut mungkin berorientasi ke satu sisi masalah (misalnya tentang reliabilitas dan efisiensi perangkat lunak), sedangkan orang lain yang menyatakan bahwa perangkat lunak itu buruk menggunakan sudut pandang yang lain lagi (aspek kegunaan dan desain). Namun, berbagai teknik dan pola implementasi tersebut memiliki sejumlah karakterisitik baku yang berlaku umum sesuai dengan kaidah perancangan berorientasi objek. Karakteristik baku ini seharusnya dapat dikuantifikasi sehingga menghasilkan seperangkat parameter sebagai alat ukur kualitas sebuah desain berorientasi objek.

Beberapa riset terdahulu telah menghasilkan alat ukur kualitas perangkat lunak yang diwujudkan dalam beberapa parameter-parameter seperti Chidamber and Kemerer's (CK) metrics suite [1], Brito e Abreu's MOOD Metrics [2] dan Baisya and Davis's QMOOD [3] dan masih banyak lagi. Namun, masih banyak kesulitan

yang didapat dalam menentukan kualitas sebuah perangkat lunak. Pertama, hasil pengukuran tunggal tidak dapat memberikan kesimpulan terhadap kualitas dari suatu perangkat lunak dan yang kedua adalah beberapa hasil pengukuran berupa angka-angka yang sulit dipahami oleh para pengembang perangkat lunak.

Untuk itu diperlukan sebuah aplikasi untuk membantu para pengembang memahami kualitas dari perangkat lunaknya. Tugas akhir ini akan membahas penerapan pengukuran kualitas perangkat lunak pada kerangka kerja Java Platform. Penyajian visualisasi dalam bentuk graph akan diterapkan untuk memudahkan pemahaman akan hasil pengukuran, sehingga memudahkan pembaca untuk memahaminya.

1.2. Tujuan

Tujuan dari pembuatan tugas akhir ini adalah membuat kakas bantu pengukuran kualitas perangkat lunak pada kerangka kerja Java Platform

1.3. Rumusan Permasalahan

Rumusan masalah yang diangkat dalam tugas akhir ini adalah sebagai berikut:

1. Bagaimana membangun sebuah kakas bantu pengukur kualitas perangkat lunak berbasis objek oriented yang bisa digunakan untuk kode pemrograman Java;
2. Bagaimana menampilkan visualisasi hasil yang baik dan mudah dimengerti;
3. Bagaimana melakukan perhitungan untuk setiap metrik yang ada; dan
4. Bagaimana mendeteksi kode yang terdapat dalam file.

1.4. Batasan Permasalahan

Permasalahan yang dibahas dalam tugas akhir ini memiliki beberapa batasan, di antaranya sebagai berikut:

1. Data uji yang akan digunakan adalah *source code* dari aplikasi dengan bahasa pemrograman Java;

2. Kakas hanya dapat memberi nilai pada setiap kode yang diuji;
3. Graph dan range yang digunakan mengacu pada paper. [4] [5] [6]

1.5. Metodologi

Langkah-langkah yang ditempuh dalam pengerjaan tugas akhir ini yaitu:

1. Studi literatur

Pada tahap ini dilakukan pengumpulan informasi mengenai pemrograman Java, Eclipse beserta algoritmanya. Mengumpulkan dan menggali informasi dan literatur yang diperlukan dalam proses perancangan dan implementasi sistem yang dibangun. Literatur yang digunakan adalah sebagai berikut:

- Konsep Cyclomatic Complexity, Size, Comment Percentage, Wighted Methods per Class, Response for Class, Coupling between Object Classesor, Depth of Inheritance Tree, dan Number of Children;
- Teori Object Oriented Programming;

.

2. Analisis dan Perancangan Sistem

Pada tahap analisis perancangan sistem dilakukan pendefinisian kebutuhan sistem untuk masalah yang sedang dihadapi. Selanjutnya, dilakukan perancangan sistem dengan beberapa tahap sebagai berikut:

a. analisis aktor yang terlibat didalam sistem;
b. perancangan proses aplikasi;
c. perancangan antar muka sistem; dan
d. perancangan diagram kelas sistem.

3. Implementasi

Pada tahap ini dilakukan pembuatan elemen perangkat lunak. Sistem yang dibuat berpedoman pada rancangan yang telah dibuat pada proses perancangan dan analisis sitem.

Perincian tahap ini adalah sebagai berikut:

a. implementasi pembangunan *call graph* berdasarkan kode program; dan
b. implementasi pemberian blok-blok berwarna pada kode program sebagai hasil dari analisis.

4. Pengujian dan evaluasi

Pada tahap ini dilakukan pengujian terhadap elemen perangkat lunak dengan menggunakan data uji beberapa kode java. Pengujian dan evaluasi perangkat dilakukan untuk mengevaluasi hasil analisis program. Tahapan-tahapan dari pengujian adalah sebagai berikut:

a. pencocokan hasil uji kakas dengan hasil uji secara manual; dan
b. pengujian kegunaan pada pengguna kakas.

5. Penyusunan buku tugas akhir

Pada tahap ini dilakukan pendokumentasian dan pelaporan dari seluruh konsep, dasar teori, implementasi, proses yang telah dilakukan, dan hasil-hasil yang telah didapatkan selama pengerjaan tugas akhir.

1.6. Sistematika Penulisan

Buku tugas akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan tugas akhir ini. Selain itu, diharapkan dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku tugas akhir terdiri atas beberapa bagian seperti berikut ini.

Bab I Pendahuluan

Bab ini berisi latar belakang masalah, tujuan dan manfaat pembuatan tugas akhir, permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penyusunan tugas akhir.

Bab II Dasar Teori

Bab ini membahas beberapa teori penunjang yang berhubungan dengan pokok pembahasan dan mendasari pembuatan tugas akhir ini.

Bab III Analisis dan Perancangan Sistem

Bab ini membahas mengenai perancangan perangkat lunak. Perancangan perangkat lunak meliputi perancangan data, arsitektur, prosesdan perancangan antarmuka pada kakas.

Bab IV Implementasi

Bab ini berisi implementasi dari perancangan perangkat lunak.

Bab V Pengujian dan Evaluasi

Bab ini membahas pengujiandengan metode pengujian subjektif untuk mengetahui penilaian aspek kegunaan (*usability*) dari perangkat lunak dan pengujian hasil analisis kakas.

Bab VI Kesimpulan

Bab ini berisi kesimpulan dari hasil pengujian yang dilakukan. Bab ini membahas saran-saran untuk pengembangan sistem lebih lanjut.

Daftar Pustaka

Merupakan daftar referensi yang digunakan untuk mengembangkan tugas akhir.

Lampiran

Merupakan bab tambahan yang berisi daftar istilah yang penting pada aplikasi ini.

[Halaman ini sengaja dikosongkan]

BAB II
DASAR TEORI

Pada bab ini akan dibahas mengenai teori-teori yang menjadi dasar dari pembuatan tugas akhir. Jaminan kualitas perangkat lunak adalah aktivitas pelindung yang diterapkan pada proses perangkat lunak. Jaminan ini meliputi :

- pendekatan manajemen kualitas,
- teknologi rekayasa perangkat lunak yang efektif (metode dan piranti),
- kajian teknik formal yang diaplikasikan pada keseluruhan proses perangkat lunak,
- strategi pengujian *multitiered* (deret bertingkat),
- kontrol dokumentasi perangkat lunak dan perubahan,
- prosedur untuk menjamin kesesuaian dengan standar pengembangan perangkat lunak,
- dan mekanisme pengukuran dan pelaporan.

Kontrol kualitas merupakan serangkaian pemeriksaan, kajian, dan pengujian yang digunakan pada keseluruhan siklus pengembangan untuk memastikan bahwa setiap produk memenuhi persyaratan yang ditetapkan. Konsep kunci kualitas kontrol adalah bahwa semua produk kerja memiliki spesifikasi yang telah ditentukan dan dapat diukur dimana kita dapat membandingkan *output* dari setiap proses.

2.1 Penelitian Terkait

Terdapat paper yang telah membahas klasifikasi kompleksitas ini sebelumnya di antaranya adalah:

2.1.1 E-Quality: A Graph Based Object Oriented Software Quality Visualization Tool

Penelitian ini dilakukan oleh Ural Erdemir dan Umut Tekin serta Feza Buzluca. Pada paper ini dibahas bahwa terdapat rentang antara kompleksitas pada setiap metrik. Rentang ini dapat

dijadikan acuan untuk menentukan nilai per metrik, namun tidak terdapat sebuah cara khusus untuk melakukan penghitungan dan penarikan kesimpulannya. Paper ini berfokus pada rentang klasifikasi yang terdapat pada setiap metrik saja.

2.1.2 Static and Dynamic Coupling and Cohesion Measures in Object Oriented Programming

Paper ini dibuat oleh Vasudha Dixit, Dr. Rajeev Vishwkarma. Paper ini hampir sama dengan paper lain secara umum yang hanya mengangkat satu topik metrik (dalam paper ini *coupling between object*). Metrik yang dipilih akan diterangkan secara rinci mulai dari cara menelaah maupun menghitungnya, hanya saja, satu metrik tidak dapat dikgunakan secara gamblang untuk mengukur satu kode program tertentu. Dibutuhkan metrik-metrik lainnya untuk menentukan hal tersebut.

2.1.3 Analyzing the Complexity of Java Programs using Object Oriented Software Metrics

Paper ini dibuat oleh Arti Chhikara dan R.S.Chhillar. Pada paper ini diterangkan perhitungan metrik-metrik yang dijadikan acuan untuk menentukan kompleksitas sebuah perangkat lunak. Paper ini juga membahas mengenai metode penhitungan satu per satu secara rinci, namun tidak terdapat tampilan atau visualisasi yang cukup mudah untuk dimengerti oleh pengguna ketika melakukan penghitungan pada metrik.

Untuk melakukan proses kontrol kulitas ini terdapat beberapa pilihan aspek pengujian antara lain melalui kode program dari sebuah aplikasi. Kode program dapat digunakan sebagai tolok ukur kualitas perangkat lunak dengan menyusuri metrik-metrik yang terdapat di dalamnya.

2.2 Kode Metrik

Dalam konteks rekayasa perangkat lunak, mengukur atau mengindikasikan kuantitatif dari ukuran atribut sebuah proses atau produk sangat penting dilakukan. Pengukuran adalah

kegiatan menentukan sebuah ukuran. Definisi metriks menurut IEEE adalah "ukuran kuantitatif dari tingkat dimana sebuah sistem, komponen atau proses memiliki atribut tertentu" [1] . Pengukuran ada jika suatu data tunggal telah dikumpulkan (misal: jumlah kesalahan yang ditemukan pada kajian sebuah modul tunggal). Metrik perangkat lunak menghubungkan pengukuran-pengukuran individu dengan banyak cara, misal rata-rata dari jumlah kesalahan yang ditemukan pada setiap kajian.

Dalam tugas akhir ini akan dibahas metrik-metrik yang dapat digunakan untuk melakukan pengukuran tersebut yaitu :

Gambar 2. 1 Metrik dan Defenisinya

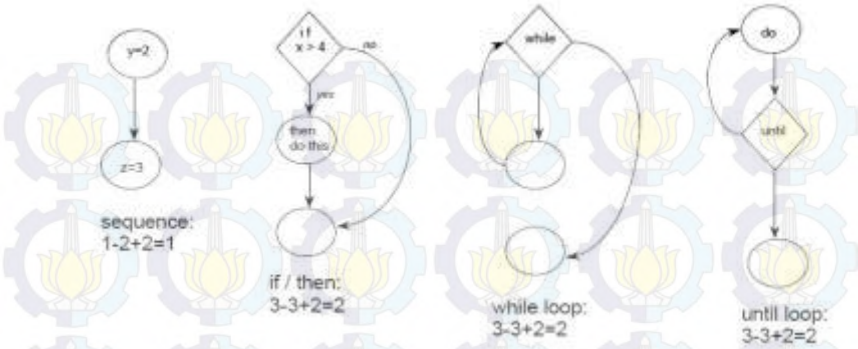
No	Nama Metrik	Defenisi
1	Depth of Inheritance Tree(DIT)	Panjang maksimum dari suatu titik hingga akar dari inherritance tree
2	Cyclomatic Complexity (CC),	Indikasi kekompleksitasan dari program dengan cara menelusuri nomor dari jalur yang independen
3	Number of Children (NoC)	jumlah subkelas dalam sebuah hirarki kelas
4	Response for Class (RFC)	Jumlah semua metode yang dipanggil sebagai respon terhadap objek diluar sebuah kelas
5	Weighted Method Complexity (WMC)	Jumlah operasi berupa fungsi-fungsi yang dapat dikerjakan oleh suatu objek.

No	Nama Metrik	Defenisi
6	Comment Percentage (CP)	Jumlah persentasi komentar yang terdapat dalam kode program yang ada
7	File Size dan Line of Code (LOC)	Metrik yang menghitung seluruh baris dalam program
8	Coupling Between Object(CBO)	Banyaknya kolaborasi untuk sebuah kelas atau jumlah hubungan kelas dengan kelas yang lain
9	Number of Subunits (NOS)	Jumlah fungsi dan prosedur setiap class

2.2.1 Siklomatik Kompleksitas

Siklomatik Kompleksitas atau *Cyclomatic complexity* atau biasa disebut juga dengan *conditional cimplexity* adalah alat pengukuran untuk mengindikasikan kekompleksitasan dari program dengan cara menelusuri nomor dari jalur yang independen melalui *source code*-nya. Dikembangkan oleh Thomas J. McCabe, Sr. Pada tahun 1976, siklomatik kompleksitas digunakan di semua fase pada daur hidup perangkat lunak, dimulai dari fase desain untuk menjaga agar perangkat lunak dapat dipercaya dan mudah untuk di tes serta lebih terorganisir [7]

Dalam menghitung Cyclomatic Complexity (CC) dalam method, pernyataan disimbolkan dengan **simpul** dan aliran program dilambangkan dengan **busur**. ***Nilai CC merupakan jumlah busur-jumlah simpul +2.*** Berikut contoh penghitungan CC :



Gambar 2. 2 Contoh perhitungan CC dari graph

Pada Gambar 2. 2, perhitungan CC didasarkan pada rumus tersebut. Pada *sequence* terdapat sebuah busur dan dua buah simpul maka untuk nilai CC nya adalah 1-2+2 yaitu 1. Untuk *until loop* terdapat masing-masing 3 buah busur dan 3 buah simpul maka nilai CC-nya adalah 2.

Metrik ini menghitung jalur keputusan tersedia dalam software untuk menentukan kompleksitas. Setiap jalur keputusan dimulai dengan salah satu pernyataan bersyarat dari daftar berikut, sehingga cukup mudah untuk mendeteksi mereka dalam kode sumber yang ada. Pernyataan itu adalah:

- `?`
- `case`
- `elseif`
- `for`
- `foreach`
- `if`
- `while`

```
public int getValue(int param1) {
    int value = 0;
    if (param1 == 0) {
     value = 4;
    }
     else {
         value = 0;
    }
    return value;
}
```

Kode 2. 1 Contoh Kode yang Memiliki Method dengan Beberapa Node Perhentian

Untuk method yang terdapat pada Kode 2. 1, terdapat dua poin keputusan : if dan else. Mengingat bahwa entry point dari method di atas secara otomatis menambahkan nilai satu, maka nilai terakhirnya sama dengan 3. Hubungan antara CC dan Resiko sendiri terdapat dalam tabel berikut ini :

Tabel 2. 1 Tingkat Resiko pada Nilai CC

CC	*Type of Procedure*	*Risk*
1-10	*A well structured and stable procedure*	*Low*
11-20	*A more complex procedure*	*Moderate*
21-50	*A complex procedure, alarming*	*High*
>50	*An error-prone, extremely troublesome, untestable procedure*	*Very high*

Kompleksitas Siklomatik yang tinggi menunjukkan prosedur kompleks yang sulit untuk dipahami, diuji dan dipelihara. Misalkan pada contoh program pada tabel 2.2, nilai dari CC yang terdapat pada program akan menentukan klasifikasi kualitas perangkat lunak tersebut :

Tabel 2. 2 Contoh Hasil Perhitungan Kompleksitas pada Kode Program berdasarkan CC

Nama Program	Nilai CC	Risk
Employments	7	Low Risk
WeightedMethod	15	Moderate
CouplingBetweenObject	28	High
OSM	328	Very High

2.2.2 Ukuran Kode

Ukuran kode merupakan metrik tradisional yang digunakan untuk mengevaluasi tingkat kemudahan oleh pengembang dan pemelihara. Seperti metrik kompleksitas siklomatik, ukuran metrik juga berpengaruh terhadap pengertian, pemeliharaan dan pengujian. Semakin rendah nilai ukur metrik ini, maka semakin mudah untuk mengerti dan memelihara kode tersebut. Ada beberapa cara untuk menghitung ukuran dari kode, diantaranya adalah jumlah baris atau Line of Code (LOC) [8] .

LOC menghitung seluruh baris dalam program. LOC dapat dikenali dengan memindai tanda-tanda akhir ‘)’, ‘}’, ‘{’ atau ‘;’. Permasalahnnya adalah LOC tidak menghitung kode yang diikuti oleh komentar sebahgai sebuah LOC.

```
Public void SetFilelOfSelected(char fill) {
      //Set fill of all selected shapes
      For (int i = 0; i<totalShapes; i++)
            If(shapes[i]. Selected())}
                  Shapes[i].SetFill(fill);
}
```

Kode 2. 2 Contoh Kode yang akan Dihitung LOC-nya

LOC sangat dipengaruhi oleh bahasa pemrograman dan gaya dari programmer. Misalkan pada potongan Kode 2. 2, terdapat jumlah baris kode sebanyak 6 baris.

Tabel 2. 3 Tingkat Resiko pada Perhitungan Nilai LOC

LOC	*Type of Size*	*Risk*
<100	*Small Size Program*	*Low*
100-200	*A well medium-low size program*	*Moderate*
200-300	*A medium big size program*	*High*
>300	*A big size program*	*Very High*

2.2.3 Persentasi Komentar

Persentasi Komentar adalah jumlah persentasi komentar yang terdapat dalam kode program yang ada. Penghitungannya adalah dengan membagi jumlah baris yang memiliki komentar dengan total jumlah baris dalam file tersebut dikalikan dengan 100%. Untuk menentukan jumlah baris komentar sendiri adalah dengan mendeteksi beberapa simbol berikut ini:

- Memiliki // di luar string
- Memiliki /* komentar */
- Memiliki /* tetapi */ tidak ditemukan pada line yang sama

```
package test;
public class Main {
/**
 * A javadoc comment.
 * @param args
 */
public static void main(String[] args) {
    System.out.println("/* This isn't a comment
*/");
    /* This is a comment */
    //Comment
    System.out.println("//Not comment");
}
//Comment
}
```

Kode 2. 3 Contoh Kode yang akan Dihitung Persentasi Komentar-nya

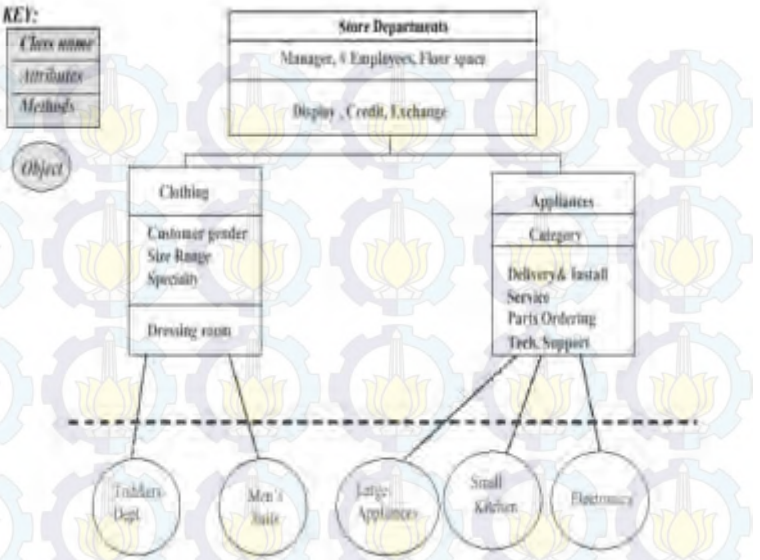
Pada Kode 2. 3 terdapat dua pasang /* komentar */ dan dua //komentar maka hasil penghitungannya adalah 4. Nilai "/* dan */" tidak dihitung.

Tabel 2. 4 Tingkat Resiko pada Perhitungan Nilai Comment Percentage

LIC	*Type of Size*	*Risk*
<90	*Well commented program*	*Low*
100-200	*A well medium-low commented program*	*Moderate*
200-300	*A medium big commented program*	*High*
>300	*A very complicated commented program*	*Very High*

2.2.4 Weighted Method Class

Weighted Method Class (WMC) adalah jumlah operasi berupa fungsi-fungsi yang dapat dikerjakan oleh suatu objek. Method didefinisikan pada kelas akan tetapi dipanggil melalui objek. Metrik ini adalah jumlah method per kelas, dimana setiap method dibebani oleh sebuah kompleksitas yang didasarkan pada tipe dari method, jumlah dari propertis method yang memengaruhi dan jumlah dari layanan yang method sediakan untuk sistem ini.



Gambar 2. 3 Contoh diagram sederhana rancangan kode

WMC adalah sebuah perhitungan dari implementasi method dalam sebuah kelas atau sejumlah kompleksitas dari method. Jumlah yang lebih besar dari method dalam kelas, lebih besar dampak potensialnya pada *children* karena *children* mewarisi semua method yang didefinisikan dalam sebuah kelas. Kelas-kelas dengan jumlah method yang besar mungkin akan mejadi alokasi yang lebih khusus, membatasi kemungkinan penggunaan kembali. Pada Gambar 2. 3, WMC dihitung dengan memakai jumlah dari methods dalam setiap class, sehingga WMC untuk Clothing_dept adalah 1 dan nilai WMC untuk Appliance_dept adalah 4 [9] .

Dalam kode cara menghitungnya adalah dengan memindai setiap baris pada code yang memiliki keywords untuk method, yaitu ‘void’.

```
package test;
public class ProjectLogger{
public static void getInfoLog (String message) {
    System.out.println("INFO : " + message);
}
public static void getErrorLog (String message) {
    System.out.println("Error : " + message);
}
//Comment
}
```

Kode 2. 4 Contoh Kode yang Menggunakan Method

Pada Kode 2. 4 nilai wmc dilihat dari jumlah total method yg terdapat pada class, jumlah method pada class di atas dapat di lihat dari keywords void. Kelas di atas memiliki 2 buah method, yakni getInfoLog dan getErrorLog. Maka jumlah WMC nya adalah 2.

Tabel 2. 5 Tingkat Resiko pada Nilai Perhitungan Weighted Method Complexity

WMC	*Type of Complexity*	*Risk*
<20	*A well structured and stable complexity*	*Low*
20-35	*A more complex program*	*moderate*
35-50	*A complex procedure, alarming*	*High*
>50	*An error-prone, extremely troublesome, untestable procedure*	*Very high*

2.2.5 Response For Class

Response for Class (RFC) adalah jumlah semua metode yang dipanggil sebagai respon terhadap objek diluar sebuah kelas. RFC juga mengukur komunikasi antar objek. Hal ini berlaku terhadap semua metode yang diakses dalam class hirarki, sehingga semakin banyak metode yang digunakan untuk merespon objek dari luar semakin kompleks dan meningkat waktu untuk melakukan pengujian. Nilai RFC merupakan nilai dari hasil

penambahan jumlah method yang terdapat pada kelas itu sendiri dengan jumlah method yang dipanggil dari kelas lain [10].

Dalam kode, cara menghitungnya adalah dengan menghitung dan menampung data method dari setiap kelas dengan mencari keyword "void", dan kemudian menampung data kelas dan method tersebut. Setelah itu dilakukan penghitugan untuk setiap pemanggilan kelas pada kelas lain serta penggunaan method yang terdapat pada kelas yang dipanggil.

```
Public class ClassA
{
        Private ClassB classB = new ClassB();
        Public void doSomething(){
                System.out.println("doSomething");
                }

        Public void doSomethingBasedOnClassB(){
                System.out.println (classB.toString());
        }
}
```

Kode 2. 5 Contoh Kode yang Tidak Memakai Setter dan Getter standar

Terdapat 4 proses pada Kode 2. 5 yaitu new ClassB(), doSomething(), doSomethingBasedOnClassB(), dan empty ClassA(). Jadi nilai RFC-nya adalah 4. Sebagai catatan, dalam penghitungan RFC, setter dan getter standard dalam kode tidak dihitung.

```
public class Country {
        private String code;
        private double vat;
        public String getCode() {
                return code;
        }
...
```

```
        public void setCode(String Code) {
                this.code=code;
        }
        public double getVat() {
                return vat;
        }
        public void setVat(double vat) {
                this.vat;
        }
}
```

Kode 2. 6 Contoh Kode yang Mengunakan Setter dan Getter

Pada Kode 2. 6, hanya bernilai 1 untuk empty class Country(). getCode, setCode, getVat, setVat tidak termasuk dalam penghitungan.

Tabel 2. 6 Tingkat Resiko pada Nilai Perhitungan Response Class

WMC	*Type of Hierarchy*	*Risk*
<12	*A well structured and stable hierarchy*	*Low*
12-15	*A more complex hierarchy*	*moderate*
15-20	*A complex hierarchy, alarming*	*High*
>20	*An error-prone, extremely troublesome, untestable hierarchy*	*Very high*

2.2.6 Number of Subunits

Number of Subunits digunakan untuk melakukan penghitungan jumlah fungsi dan prosedur setiap class. Cara menghitungnya adalah dengan memindai jumlah prosedur(method) yang ada dengan keywords void dan memindai jumlah fungsi yang ada berdasarkan keywords return [10].

```
package test;

public class Util{
.....
```

```
  public static boolean isEmptyString (String value) {
    if(StringUtils.isEmpty(value)){
      return true;
    } else{
      Return false;
    }
  }
  public static void getErrorLog (String message) {
    System.out.println("Error : " + message);
  }
}
```

Kode 2. 7 Contoh Kode yang Menggunakan Fungsi dan Prosedur

Pada Kode 2. 7, hasil perhitungan untuk *Number of Subunits*-nya adalah 2. Karena pada potongan kode tersebut terdapat 1 buah fungsi yaitu isEmptyString dan 1 buah prosedur yaitu getErrorLog.

Tabel 2. 7 Tingkat Resiko pada Nilai Perhitungan Number of Subunits

WMC	*Type of procedure and function*	*Risk*
<12	*A well structured and stable procedure and function*	*Low*
12-15	*A more complex procedure and function*	*moderate*
15-20	*A complex procedure and function, alarming*	*High*
>20	*An error-prone, extremely troublesome, untestable procedure and function*	*Very high*

2.2.7 Coupling Between Object Classes

Coupling Between Object Classes (CBO) adalah banyaknya kolaborasi untuk sebuah kelas atau jumlah hubungan kelas dengan kelas yang lain. Seiring CBO meningkat, kemungkinan besar kemampuan penggunaan kembali kelas akan menurun. Nilai CBO tinggi juga memersulit perubahan dan pengujian yang akan

terjadi ketika perubahan dilakukan [6]. Cara menghitungnya adalah pada string yang merupakan isi dari inputan akan di list terlebih dahulu seluruh class yang ada, kemudian dilakukan pemeriksaan apakah class tersebut merupakan *child class* dari kelas lain, kemudian dilakukan pula pemeriksaan apakah pada kelas tersebut ada dilakukan pemanggilan kelas lain dengan keyword "*new*" dan memeriksa apakah yang dipanggil merupakan data dari class yang ada atau bukan (kelas yang mungkin merupakan bagian dari library yang ada).

```
package test;

public class UserManager{
  public List getUser (String username, String
userGroup) {
      DBManager dbManager = new DBManager();
      return dbManager.getUser(username, userGroup);
  }
  Public List getUser (boolean errFlag) {
      List lstUser = new ArrayList();
      for(int i = 0; i < 10; i++){
         lstUser.add("user"+i);
      }
  return lstUser;
  }
}
```

Kode 2. 8 Contoh Kode yang Menggunakan Pemanggilan Kelas

Pada Kode 2. 8 hasil perhitungannya adalah 1. Pada potongan kode tersebut hanya terdapat 1 kali pemanggilan kelas lain yang bukan merupakan bagian dari kelas pada *library*.

Tabel 2. 8 Tingkat Resiko pada Nilai Perhitungan Coupling Between Object

WMC	*Type of procedure and function*	*Risk*
<10	*A well structured and stable coupling*	*Low*
10-20	*A more complex coupling*	*moderate*
20-30	*A complex coupling, alarming*	*High*

WMC	*Type of procedure and function*	*Risk*
>30	*An error-prone, extremely troublesome, untestable coupling*	*Very high*

2.2.8 Depth of Inheritance Tree

Depth of Inheritance Tree (DIT) adalah metrik yang berupa panjang maksimum dari suatu titik hingga akar dari inherritance tree. Seiring pertumbuhan DIT, kemungkinan besar kelas-kelas tingkat rendah akan mewarisi banyak metode. Hal ini mengarah pada kesulitan potensial ketika mencoba memerediksi tingkah laku sebuah kelas. Sebuah hirarki yang dalam (nilai DIT besar) juga mengarah pada semakin besarnya kompleksitas perancangan [11].

```
class Box {
      double width;
      double height;
      double depth;
      Box() {
      }
      Box(double w, double h, double d) {
            width = w;
            height = h;
            depth = d;
      }
      void getVolume() {
   System.out.println("Volume is : " + width * height
*
depth);
      }
}
public class MatchBox extends Box {

      double weight;
      MatchBox() {
      }
      MatchBox(double w, double h, double d, double
m) {
            super(w, h, d);
```

```
            weight = m;
        }
        public static void main(String args[]) {
        MatchBox mb1 = new MatchBox(10, 10, 10, 10);
                mb1.getVolume();
System.out.println("width of MatchBox 1 is " +
mb1.width);
System.out.println("height of MatchBox 1 is " +
mb1.height);
System.out.println("depth of MatchBox 1 is " +
mb1.depth);
System.out.println("weight of MatchBox 1 is " +
mb1.weight);
        }
    }
```

Kode 2. 9 Contoh kode yang Menggunakan Kelas yang Memiliki Turunan

Cara menghitung DIT adalah dengan menampung data semua nama class yang ada pada file inputan, kemudian memeriksa apakah pada class tersebut terdapat inheritance dengan mencari keywords "extends". Apabila kelas tersebut merupakan turunan dari kelas lain, maka nilai dari DIT akan ditambah.

Pada Kode 2. 9 terdapat sebuah kelas yaitu Class MatchBox merupakan turunan dari kelas Box. Kedua kelas tersebut masih termasuk ke dalam Low complexity code. Semakin besar nilai DIT maka akan semakin kompleks kode program tersebut. Kode program yang kompleks beresiko lebih tinggi dalam pengembangan serta pemakaiannya.

Tabel 2. 9 Tingkat Resiko pada Nilai Perhitungan Depth of Inheritance per Class

WMC	*Type of Inheritance*	*Risk*
<3	*A well structured and stable inheritance*	*Low*
3-5	*A more complex inheritance*	*moderate*
5-7	*A complex inheritance, alarming*	*High*
>7	*An error-prone, extremely troublesome,*	*Very high*

WMC	*Type of Inheritance*	*Risk*
	untestable inheritance	

2.2.9 Number of Children

Number of Children (NOC) adalah jumlah subkelas dalam sebuah hirarki kelas. NOC merupakan indikator besarnya pengaruh sebuah kelas terhadap desain sistem secara keseluruhan. Semakin besar nilai NOC, semakin besar pula potensi ketidakcocokan subclass dengan abstraksi pada kelas induk.

```
public class Parent {
      private String output = "hallo";
      public void print(){
            System.out.println(output);
      }
public class Child extends Parent {
      private String output = "child";
}
public class Child2 extends Parent {
      private String output="child2";
}
```

Kode 2. 10 Contoh Kode yang Memiliki Anak Kelas

NOC dihitung dengan cara membaca data inputan dan menampungnya dalam string, kemudian dari data string tersebut akan dipindai kelas data dan ditampung dalam list [9]. Setiap kelas akan diperiksa apakah merupakan anak dari kelas lain atau bukan dengan menggunakan keyword 'extends'. Apabila kelas tersebut merupakan kelas anak, maka cari kelas induknya pada daftar yang telah ditampung, kemudian tambahkan nilai 1 untuk setiap anak kelas yang memiliki induk. Apabila ditemukan anak kelas lain yang memiliki induk yang sama maka nilainya juga akan ditambahkan.

Pada kode Kode 2. 10 terdapat dua anak kelas di dalamnya, kode tersebut yaitu Child yang memiliki kelas parent Parent dan Child2 yang memiliki kelas parent Parent.

Tabel 2. 10 Tingkat Resiko pada Nilai Perhitungan Number of Children per Class

WMC	*Type of Inheritance*	*Risk*
<3	*A well structured and stable inheritance*	*Low*
3-5	*A more complex inheritance*	*moderate*
5-7	*A complex inheritance, alarming*	*High*
>7	*An error-prone, extremely troublesome, untestable inheritance*	*Very high*

2.3 Eclipse

Eclipse merupakan suatu *Integrated Development Environment* (IDE) yang bersifat ekstensibel. Tujuan utamanya ialah memberikan layanan untuk mengatur suatu kumpulan alat yang saling bekerja sama dalam mendukung tugas-tugas pemrograman. Bahasa pemrograman utama Eclipse ialah Java [12]. Kelebihan Eclipse daripada Software yang lain antara lain:

- Multi-platform: Bisa dijalankan di Microsoft Windows, Linux, Solaris, AIX, HP-UX dan Mac OS X.
- Mulit-language: pada dasarnya Eclipse dikembangkan dengan bahasa pemrograman Java, selain itu Eclipse juga mendukung pengembangan aplikasi berbasis bahasa pemrograman lainnya, seperti C/C++, Cobol, Python, Perl, PHP, dan lain sebagainya
- Multi-role: Selain sebagai IDE untuk pengembangan aplikasi, Eclipse pun bisa digunakan untuk aktivitas dalam siklus pengembangan perangkat lunak, seperti dokumentasi, test perangkat lunak, pengembangan web, dan lain sebagainya.

Karena fleksibilitas Eclipse yang cukup tinggi, antara lain bisa dijalankan di semua operasi system/OS dan juga open source, penggunanya sangat populer di kalangan pengembang saat ini .

2.4 NetBeans

Netbeans adalah sebuah aplikasi Integrated Development Environment (IDE) yang berbasiskan Java dari Sun Microsystems yang berjalan di atas swing. Swing merupakan sebuah teknologi Java untuk pengembangan aplikasi dekstop yang dapat berjalan pada berbagai macam platform seperti windows, linux, Mac OS X dan Solaris. Sebuah IDE merupakan lingkup pemrograman yang di integrasikan ke dalam suatu aplikasi perangkat lunak yang menyediakan Graphic User Interface (GUI), suatu kode editor atau text, suatu compiler dan suatu debugger [13].

Netbeans juga dapat digunakan progammer untuk menulis, meng-compile, mencari kesalahan dan menyebarkan program netbeans yang ditulis dalam bahasa pemrograman java namun selain itu dapat juga mendukung bahasa pemrograman lainnya dan program ini pun bebas untuk digunakan dan untuk membuat professional dekstop, enterprise, web, and mobile applications dengan Java language, C/C++, dan bahkan dynamic languages seperti PHP, JavaScript, Groovy, dan Ruby. Saat ini netbeans memiliki 2 produk yaitu Platform Netbeans dan Netbeans IDE. Platform Netbeans merupakan framework yang dapat digunakan kembali (reusable) untuk menyederhanakan pengembangan aplikasi desktop dan Platform NetBeans juga menawarkan layanan-layanan yang umum bagi aplikasi dekstop, mengijinkan pengembang untuk fokus ke logika yang spesifik terhadap aplikasi. Fitur fitur yang terdapat dalam netbeans antara lain:

1. Smart Code Completion: untuk mengusulkan nama variabel dari suatu tipe, melengkapi keyword dan mengusulkan tipe parameter dari sebuah method.
2. Bookmarking: fitur yang digunakan untuk menandai baris yang suatu saat hendak kita modifikasi.
3. Go to commands: fitur yang digunakan untuk jump ke deklarasi variabel, source code atau file yang ada pada project yang sama.

4. Code generator: jika kita menggunakan fitur ini kita dapat meng-generate constructor, setter and getter method dan yang lainnya.
5. Error stripe yang menandai baris yang eror dengan memberi *highlight* merah.

BAB III
ANALISIS DAN PERANCANGAN SISTEM

Bab ini membahas tahap analisis permasalahan dan perancangan dari sistem yang akan dibangun. Analisis permasalahan membahas permasalahan yang yang diangkat dalam pengerjaan tugas akhir. Analisis kebutuhan mencantumkan kebutuhan-kebutuhan yang diperlukan perangkat lunak. Selanjutnya dibahas mengenai perancangan sistem yang dibuat. Pendekatan yang dibuat dalam perancangan ini adalah pendekatan berorientasi objek. Perancangan direpresentasikan dengan diagram UML (*Unified Modelling Language*).

3.1. Analisis

Tahap analisis dibagi menjadi beberapa bagian antara lain cakupan permasalahan, deskripsi umum sistem, kasus penggunaan sistem, dan kebutuhan perangkat lunak.

3.1.1. Analisis Permasalahan

Permasalahan utama yang diangkat dalam pembuatan tugas akhir ini adalah bagaimana melakukan penghitungan metrik untuk mengukur tingkat resiko suatu kode program yang berbahasa Java. Dan permasalahan berikutnya adalah bagaimana menampilkan hasil perhitungan tersebut dalam bentuk chart sehingga dapat dipahami lebih mudah oleh penggunanya.

Metrik-metrik yang digunakan untuk melakukan penghitungan nilai resiko program pada tugas akhir ini adalah Depth of Inheritance Tree(DIT), Cyclomatic Complexity (CC), Number of Children (NoC), Response for Class (RFC), Weighted Method Complexity (WMC), Comment Percentage (CP), File Size, Coupling Between Object(CBO), Number of Subunits (NOS), dan Line of Code (LOC). Pada tugas akhir ini pemeriksaan dan pengukuran nilai resiko difokuskan pada Bahasa pemrograman Java karena Bahasa pemrograman ini merupakan

salah satu Bahasa pemrograman utama dalam Object Oriented Programming dan sangat banyak digunakan.

3.1.2. Deskripsi Umum Sistem

Sistem yang akan dibuat berupa program yang akan membantu melakukan penghitungan terhadap metrik-metrik yang ingin dihitung. Program ini nantinya akan menampilkan data tiap metrik dalam bentuk chart sehingga dapat dibandingkan untuk setiap metriknya. Pengguna dapat memilih metrik yang ingin dilakukan penghitungan, untuk beberapa metrik akan dilakukan penghitungan setiap classnya dan beberapa metrik melakukan perhitungan untuk keseluruhan file java yang diberikan sebagai inputan. Hasil dari perhitungan tingkat resiko dari inputan yang diberikan akan ditampilkan dalam file .pdf, di mana hasilnya akan ditunjukkan dalam bentuk barchart dan teks berupa keterangan.

Diharapkan dengan adanya program ini, pengguna dapat melakukan penghitungan dan melakukan manajemen resiko untuk mengurangi tingkat resiko dan kompleksitas dari file yang diberikan sebagai inputan.

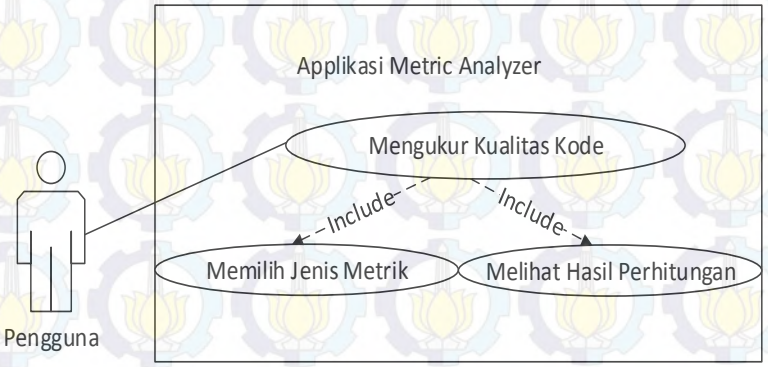
3.1.3. Aktor

Aktor mendefinisikan entitas-entitas yang terlibat dan berinteraksi langsung dengan sistem. Entitas ini bisa berupa manusia maupun sistem atau perangkat lunak yang lain. Aktor yang terdapat pada sistem ini hanya memiliki sebuah peran yaitu sebagai pengguna. Pengguna perangkat ini adalah pengembang applikasi java yang ingin melakukan perhitungan nilai kompleksitas resiko dari program yang dibangunnya.

3.1.4. Kasus Penggunaan

Berdasarkan analisis spesifikasi kebutuhan fungsional dan analisis aktor dari sistem dibuat kasus penggunaan sistem. Kasus-kasus penggunaan dalam sistem ini akan dijelaskan secara rinci pada subbab ini. Kasus penggunaan digambarkan dalam sebuah diagram kasus penggunaan. Diagram kasus penggunaan

dapat dilihat pada Gambar 3. 1 dan Tabel 3. 1 berisi penjelasan dari setiap kasus penggunaan.



Gambar 3. 1 Diagram Kasus Penggunaan

Tabel 3. 1 Daftar Kode Diagram Kasus Penggunaan

Kode Kasus Penggunaan	Nama
UC-0001	Mengukur Kualitas Kode

Pada kasus penggunaan ini, pengguna memberikan input berupa berkas dengan ekstensi .java atau .txt yang akan diperiksa dan dilakukan penghitungan nilai metriknya. Pengguna akan menekan tombol *choose file* guna menentukan alamat dari berkas yang akan diperiksa dan memilih jenis metrik yang akan dihitung. System akan melakukan perhitungan nilai metrik dan menampilkan data hasil perhitungan.

Tabel 3. 2 Skenario *Use Case* Mengukur Kualitas Code

Nama	Skenario Use Case Mengukur Kualitas Code
Kode	UC-0001
Deskripsi	Aktor memilih file yang akan dijadikan inputan untuk dilakukan perhitungan nilai metric. Kemudian aktor akan memilih data metric yang ingin dihitung.
Pemicu	Pengguna menekan tombol *choose file* dan mencentang tipe metrik yang ingin dihitung, kemudian menekan tombol *calculate*.
Aktor	Pengguna
Kondisi Awal	Form Utama applikasi terbuka
Skenario Normal	1. Pengguna menekan tombol choose file. 2. Pengguna memilih file .java atau .txt yang berisi code yang ingin dihitung. 3. Pengguna memilih jenis metrik yang dihitung. 4. Pengguna menekan tombol calculate 5. Sistem akan membaca file dan mengurai code yang ada pada file 6. Sistem melakukan perhitungan berdasarkan metrik yang dipilih. 7. Sistem akan menampilkan hasil perhitungan.
Kondisi Akhir	Sistem melakukan perhitungan data metrik dan resiko dari *code* yang terdapat pada file
Skenario Alternatif 1	1. Pengguna langsung menekan tombol *calculate* tanpa memilih inputan file ataupun memilih data metric yang diinginkan 2. Sistem akan menampilkan pesan error meminta pengguna memilih inputan file terlebih dahulu
Kondisi Akhir	Sistem menampilkan form utama dan meminta pengguna memilih inputan file
Skenario Alternatif 2	1. Pengguna memilih file inputan 2. Pengguna menekan tombol *calculate* tanpa memilih data metrik yang ingin dihitung 3. Sistem akan menampilkan pesan error, meminta pengguna memilih data metrik yang ingin dihitung terlebih dahulu.

Nama	Skenario Use Case Mengukur Kualitas Code
Kondisi Akhir	Sistem menampilkan form utama dan meminta pengguna memilih tipe metrik yang ingin dihitung

Gambar 3. 2 Diagram Urutan Mengukur Kualitas Kode

Pada gambar 3.2, dijelaskan bahwa pembangunan kakas bantu ini dirancang dengan membagi kode program ke dalam empat package(kelompok) terpisah. Keempat kelompok tersebut adalah:

1. Package GUI

Package GUI yaitu kelompok yang berfungsi sebagai penanganan tampilan applikasi dengan pengguna. Pada package ini terdapat class MainForm yang merupakan tampilan utama sebagai halaman interaksi antara pengguna dengan sistem. pada halaman ini pengguna dapat memilih source code yang akan diperiksa, memilih jenis metrik yang ingin dihitung, dan memberikan perintah pada sistem untuk melakukan penghitungan nilai metrik.

2. Package Controller

Package Controller yaitu kelompok yang berfungsi sebagai perantara komunikasi antara proses penghitungan data yang dilakukan oleh sistem dengan tampilan utama. kelompok ini berguna sebagai pemanggil nilai hasil perhitungan dari metrik yang dipilih untuk kemudian di tampilkan.

3. Package Entity

Package Entity yaitu kelompok yang berfungsi sebagai pengolah data pada sistem. Class-class yang terdapat pada kelompok ini berfungsi untuk melakukan proses perhitungan dari masing-masing metrik. Seluruh pengolahan nilai metrik dan penentuan tingkat resiko yang dimiliki masing-masing metrik dilakukan pada class-class yang terdapat pada kelompok ini.

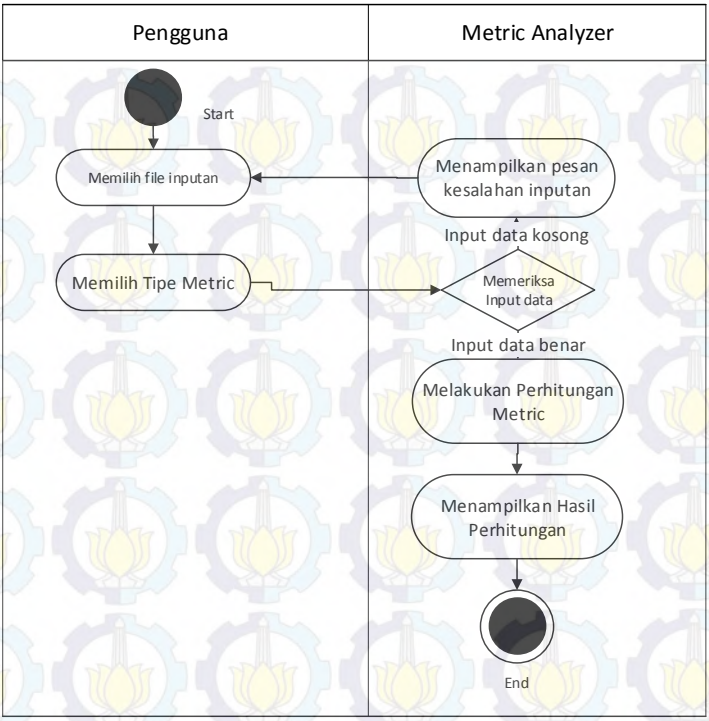
4. Package Model

Package Model yaitu kelompok yang berfungsi sebagai tipe data penampung untuk mempermudah pengiriman hasil perhitungan dari masing-masing metrik. Class-class yang terdapat pada kelompok ini merupakan class yang digunakan untuk menampung data hasil perhitungan ataupun data yang akan digunakan pada hasil akhir(file pdf). Kelompok model ini digunakan dengan pertimbangan untuk membuat pengiriman data lebih mudah dan lebih rapi. Class yang terdapat pada kelompok

ini hanya memiliki atribut yang diperlukan sesuai dengan tipe data yang akan digunakan serta setter dan getter standard.

Alur proses yang dilakukan pada program ini berdasarkan pembagian kelompok di atas adalah sbb:

1. Pengguna akan memilih data inputan (source code) yang akan dihitung dan memilih jenis metrik yang ingin dihitung melalui Package GUI.
2. Package GUI akan mengirimkan alamat inputan pada class controller dan meminta package controller nilai hasil perhitungan sesuai dengan metrik yang dipilih.
3. Package controller akan mengirim alamat inputan dan meminta package entity untuk melakukan perhitungan nilai dari metrik yang telah dipilih.
4. Package Entity akan melakukan perhitungan nilai metrik dari inputan yang diberikan, dan kemudian menampung hasil perhitungan tersebut ke dalam bentuk Package model.
5. Nilai hasil perhitungan yang ditampung dalam bentuk package model kemudian akan dikirimkan oleh package entity kepada
2. package controller.
1. Package controller akan mengirimkan nilai hasil perhitungan dalam bentuk package entity kepada class yang terdapatpada package GUI untuk kemudian diolah dan disusun.
2. Data yang telah disusun kemudian akan dikirimkan ke package controller kembali untuk dibentuk kedalam hasil akhir (file pdf)
3. dan ditampilkan kembali pada pengguna.

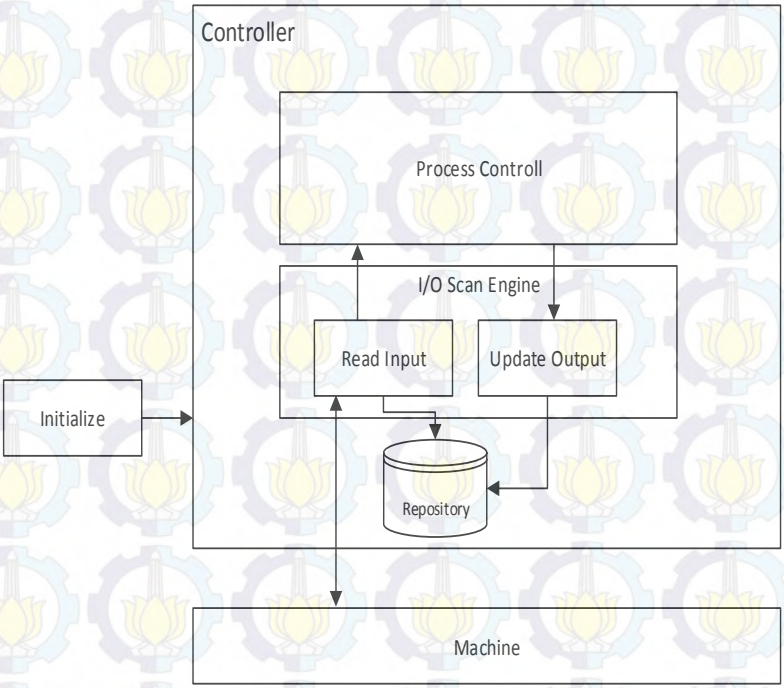


Gambar 3. 3 Diagram Aktivitas Mengukur Kualitas Kode

Pada kasus penggunaan ini, sistem menerima input berupa file .java atau .txt yang akan diperiksa dan perintah untuk memproses kode pada file. Pengguna akan memilih data metric yang ingin dihitung. Setelah itu system akan melakukan perhitungan nilai *metric* dan resiko dari *code* yang terdapat pada file inputan berdasarkan pilihan *metric* yang dicentang oleh pengguna. Spesifikasi kasus penggunaan ini dapat dilihat pada tabel skenario *use case* pada Tabel 3. 2. Pada kasus penggunaan ini, system akan menerima perintah dari pengguna untuk melakukan perhitungan metric kompleksitas dan nilai resiko dari *code* yang terdapat pada inputan.

3.1.5. Spesifikasi Kebutuhan Perangkat Lunak

Bagian ini berisi semua kebutuhan perangkat lunak yang diuraikan secara rinci dalam bentuk diagram kasus, diagram urutan, dan diagram aktivitas. Masing-masing diagram menjelaskan perilaku atau sifat dari sistem ini. Kebutuhan perangkat lunak dalam sistem ini mencakup kebutuhan fungsional saja. Pada bab ini juga dijelaskan tentang spesifikasi terperinci pada masing-masing kebutuhan fungsional. Rincian spesifikasi dari kasus penggunaan disajikan dalam bentuk arsitektur sistem berikut :



Gambar 3. 4 Diagram Arsitektur Sistem

3.1.5.1. Kebutuhan Fungsional

Kebutuhan fungsional berisi proses-proses yang harus dimiliki sistem. Kebutuhan fungsional mendefinisikan layanan yang harus disediakan dan reaksi sistem terhadap masukan atau pada situasi tertentu. Daftar kebutuhan fungsional dapat dilihat pada Tabel 3. 3.

Tabel 3. 3 Daftar Kebutuhan Fungsional Perangkat Lunak

Kode Kebutuhan	Kebutuhan Fungsional	Deskripsi
F-0001	Memilih Metrik yang akan dihitung	Pengguna dapat memilih tipe metrik yang akan dihitung.
F-0002	Menampilkan Hasil Perhitungan	Pengguna dapat menampilkan data hasil dari perhitungan metrik.

3.1.5.2. Kebutuhan Non Fungsional

Kebutuhan non fungsional berisi batasan layanan yang dimiliki sistem. Layanan ini bias berupa batasan waktu, batasan pengembangan proses standarisasi, dan lain-lain. Daftar kebutuhan non fungsional dapat dilihat pada Tabel 3. 4.

Tabel 3. 4 Daftar Kebutuhan Non Fungsional Perangkat Lunak

Kode Kebutuhan	Kebutuhan Non Fungsional	Deskripsi
NF-0001	Memiliki visualisasi yang mudah dimengerti	Pengguna memahami tampilan aplikasi serta hasilnya dengan baik
NF-0002	Memiliki waktu penghitungan metrik yang relatif cepat	Pengguna dapat memperoleh hasil penghitungan dengan waktu yang relatif cepat

3.2.Perancangan Sistem

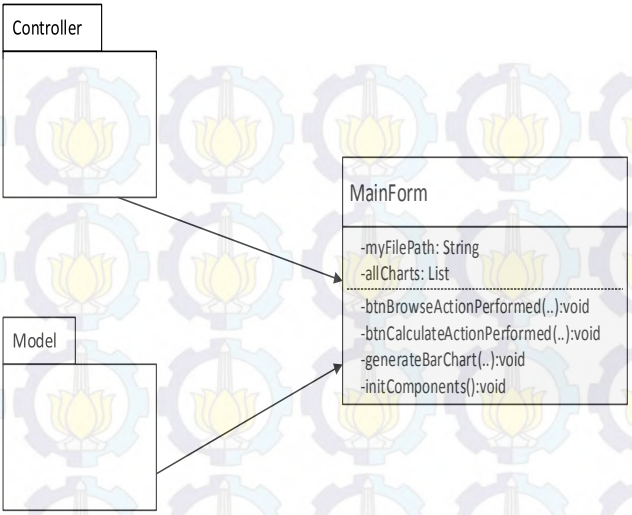
Penjelasan tahap perancangan perangkat lunak dibagi menjadi beberapa bagian yaitu perancangan diagram kelas, perancangan proses analisis, dan perancangan antarmuka.

3.2.1. Perancangan Diagram Kelas

Perancangan diagram kelas berisi rancangan dari kelas-kelas yang digunakan untuk membangun sistem. Pada subbab ini, hubungan dan perilaku antar kelas digambarkan dengan lebih jelas. Tiga lapisan pada arsitektur ini terdiri dari lapisan antarmuka, kontrol, dan data. Lapisan kontrol merupakan penghubung antara lapisan antarmuka dengan lapisan data. Subbab ini dibagi menjadi tiga bagian, yaitu diagram kelas untuk lapisan antarmuka, kontrol, Entity dan model.

3.2.1.1. Diagram Kelas Lapisan Antarmuka

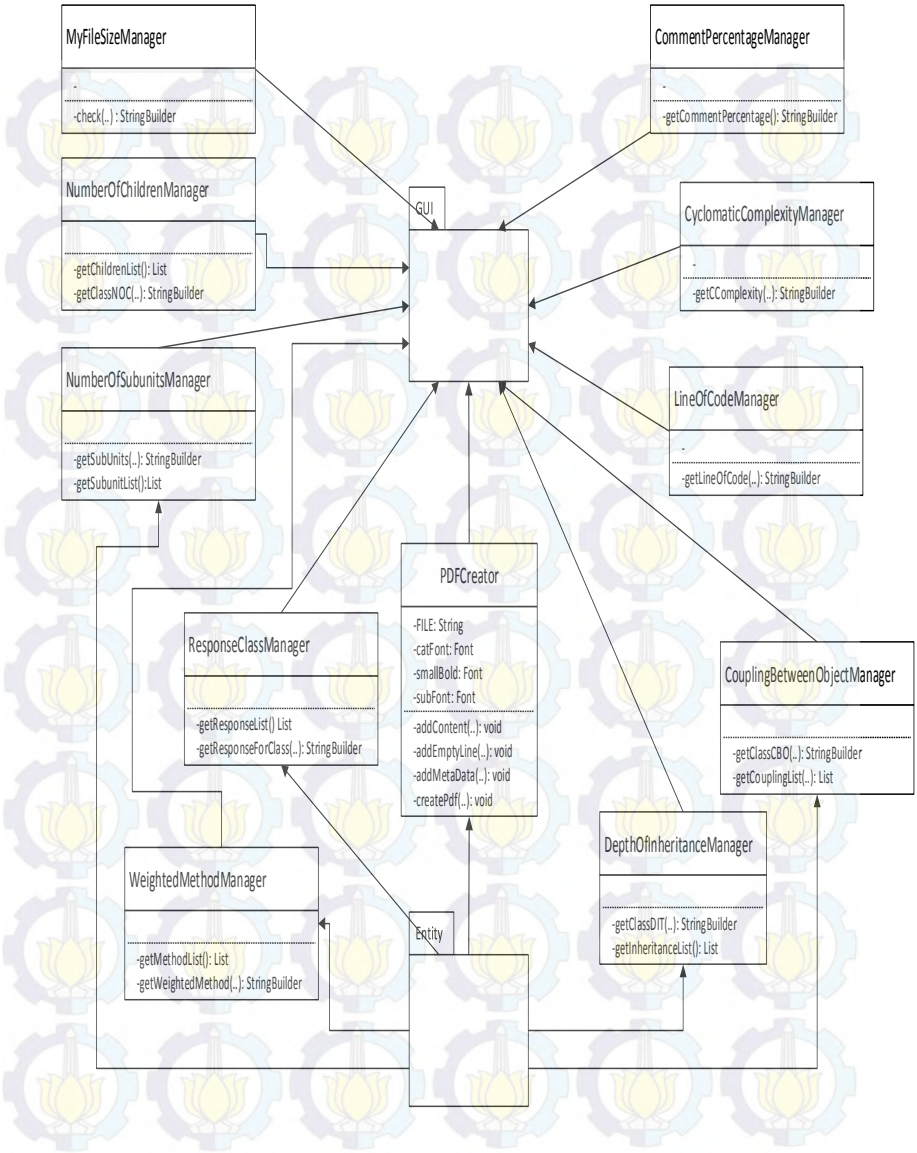
Pada lapisan ini hanya terdapat sebuah kelas, yakni kelas *MainForm,* yakni kelas utama yang juga merupakan lapisan penghubung sistem dengan actor pengguna. Kelas ini merupakan kelas yang berinteraksi langsung dengan pengguna sistem. Kelas ini terhubung dengan lapisan kontrol dan lapisan model.



Gambar 3. 5 Diagram Kelas Lapisan Antarmuka

3.2.1.2. Diagram Kelas Lapisan Kontrol

Kelas-kelas penyusun lapisan kontrol dapat dilihat pada Gambar 3. 6. Kelas-kelas tersebut adalah *MyFileSizeManager, CommentPercentageManager, CouplingBetweenObjectManager, NumberOfChildrenManager, PDFCreator, LineOfCodeManager, CyclomaticComplexityManager, NumberOfSubunitsManager, DepthOfInheritanceManager, ResponseClassManager,* dan *WeightedMethodManager*. Kelas-kelass pada lapisan kontrol ini berguna untuk menghubungkan lapisan entity dengan lapisan antarmuka.



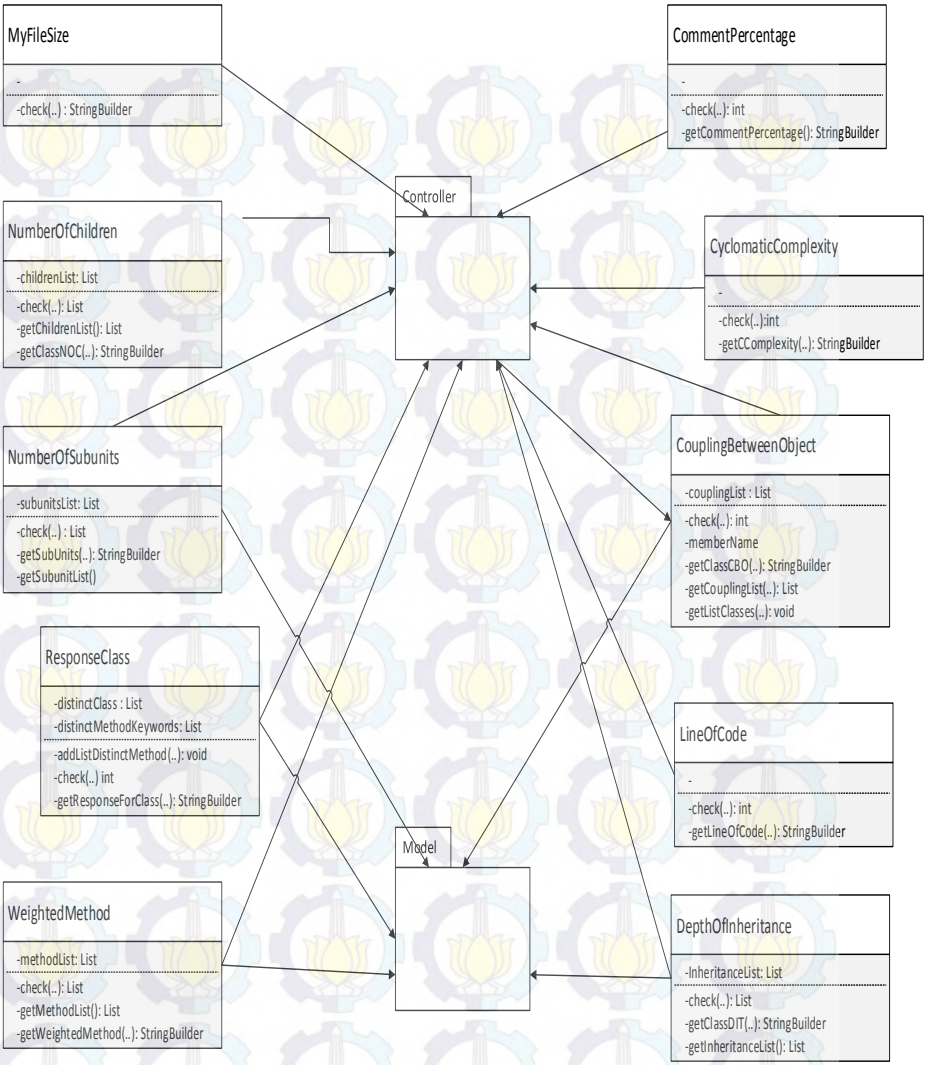
Gambar 3. 6 Diagram Kelas Lapisan Kontrol

3.2.1.3. Diagram Kelas Lapisan Entity

Kelas–kelas penyusun lapisan entity dapat dilihat pada Gambar 3. 7. Kelas-kelas tersebut antara lain kelas *CommentPercentage, CouplingBetweenObject, NumberOfChildren, LineOfCode, CyclomaticComplexity, MyFileSize, NumberOfSubunits, DepthOfInheritance , ResponseClass,* dan *WeightedMethod*. Kelas-kelas pada lapisan entity ini adalah kelas-kelas yang berfungsi untuk melakukan kalkulasi setiap metrik yang disediakan pada applikasi ini.

Setiap penamaan kelas pada lapisan ini menunjukkan metrik yang dihitung pada inputan. *CommentPercentage* merupakan kelas yang berguna untuk melakukan perhitungan jumlah *comment* yang terdapat pada inputan. Kelas *CouplingBetweenObject* berfungsi untuk melakukan perhitugan metrik jumlah *coupling* yang terjadi antar object pada inputan. Kelas *NumberOfChildren* berfungsi untuk melakukan penghitungan jumlah *Children* dari kelas yang ada pada inputan. Kelas *LineOfCode* berguna untuk menghitung jumlah baris *code. CyclomaticComplexity* berfungsi menghitung jumlah kompleksitas dari *node* yang ada pada kelas.

Kelas *DepthOfInheritance* berguna untuk melakukan penghitungan kedalaman *Inheritance* pada setiap kelas di inputan. Kelas *MyFileSize* berguna untuk menghitung *size* dari inputan. Kelas *NumberOfSubunits* berguna untuk melakukan penghitungan jumlah fungsi dan prosedur yang terdapat pada inputan. Kelas *ResponseClass* berfungsi untuk menghitung jumlah method yang ada pada setiap kelas dan jumlah method dari kelas lain yang digunakannya. Dan Kelass *WeightedMethod* berguna untuk menghitung jumlah kompleksitas dari method yang terdapat pada setiap kelas.

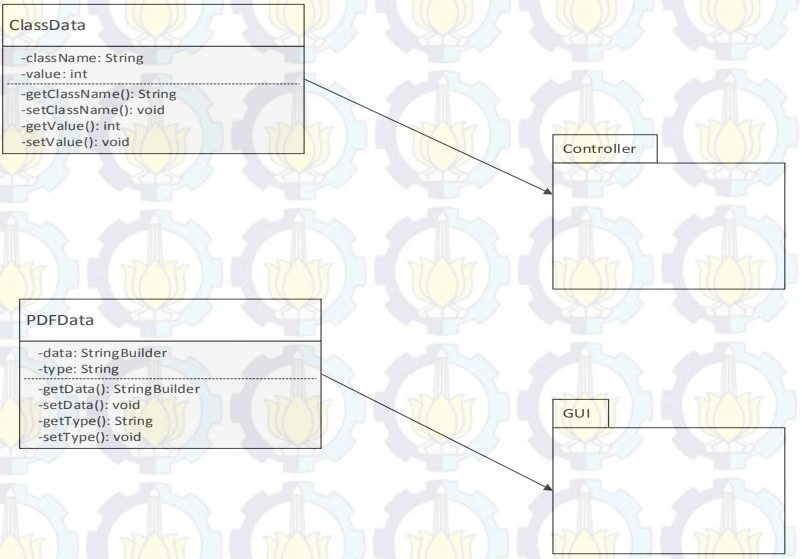


Gambar 3. 7 Diagram Kelas Lapisan Entity

3.2.1.4. Diagram Kelas Lapisan Model

Gambar 3. 8 menunjukkan diagram untuk kelas lapisan entity. Lapisan data berisi kelas-kelas yang merepresentasikan struktur data yang dibutuhkan dalam melakukan perhitungan metrik dan pembentukan hasil pdf akhir untuk menampilkan hasil perhitungan. Kelas-kelas tersebut adalah kelas *ClassData* dan *PDFData*.

Kelas *ClassData* merupakan kelas model yang digunakan sebagai penampung data hasil perhitungan metrik. Kelas ini berisi atribut-atribut yang dibutuhkan untuk menampung data nama classs dan nilai dari metrik yang dihitung. Kelas *PDFCreator* merupakan kelas yang digunakan untuk melakukan pengolahan data yang ada untuk dibentuk ke dalam file .pdf. Kelas PDFData digunakan untuk menampung data teks yang nantinya akan ditampilkan pada pdf hasil akhir.



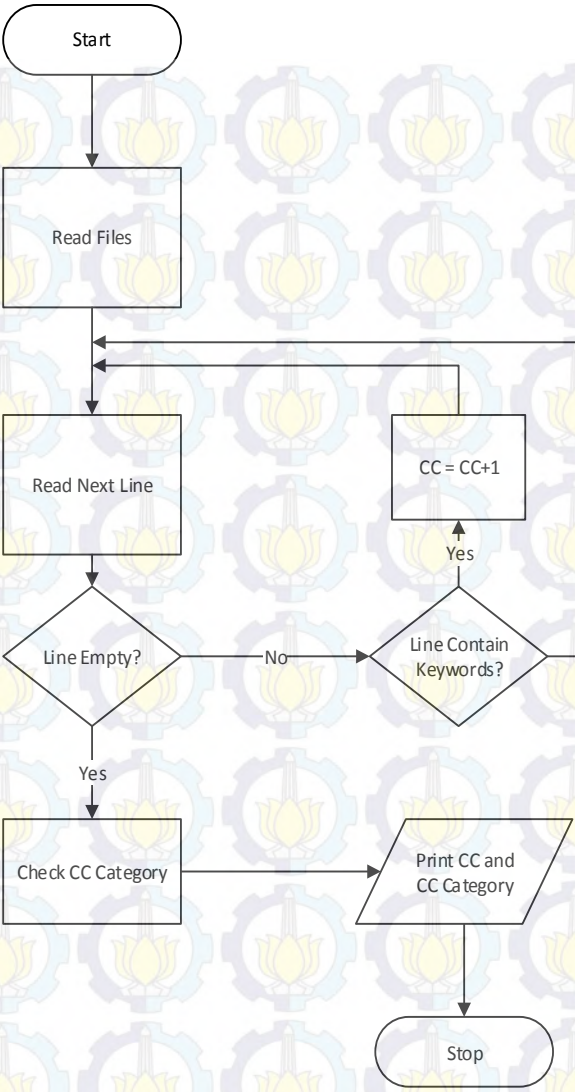
Gambar 3. 8 Diagram Kelas Lapisan Model

3.2.2. Perancangan Proses Penghitungan Metrik

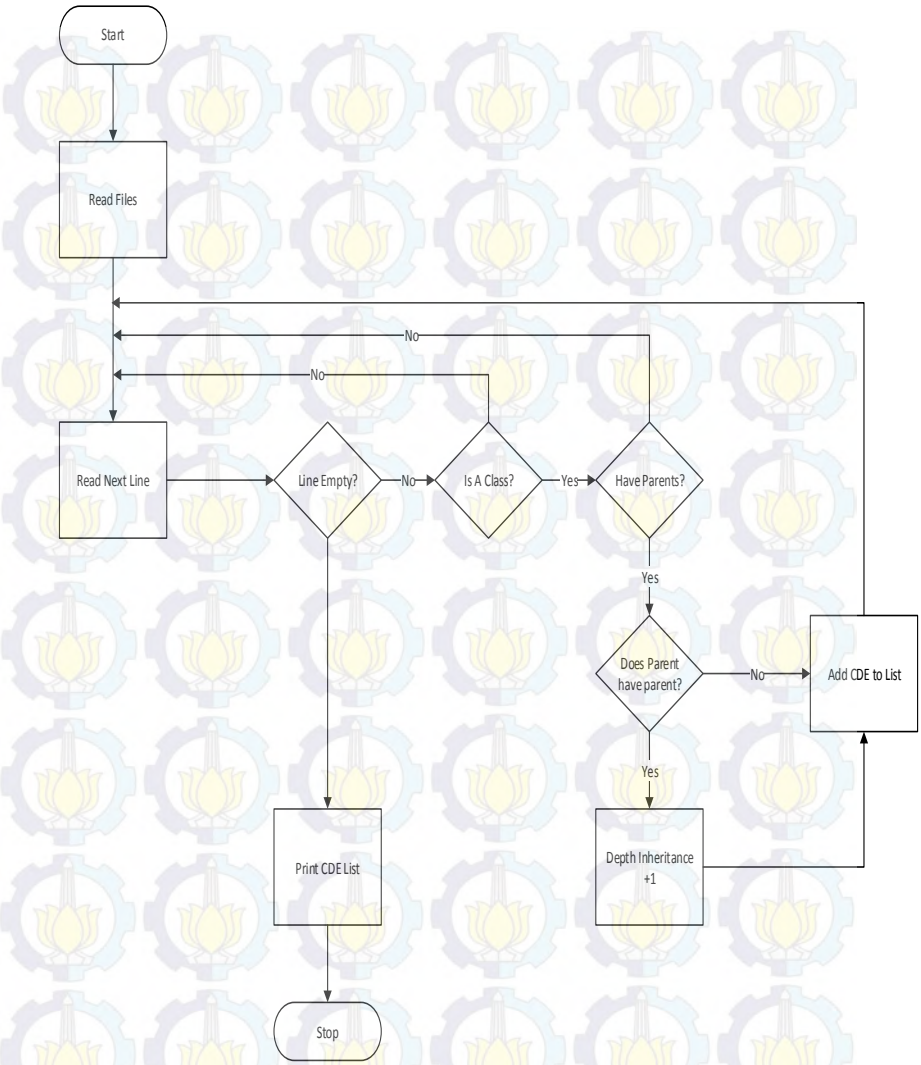
Pada bagian ini akan dijelaskan proses yang terjadi pada program dalam menghitung nilai metrik. Proses penghitungan metrik ini menggunakan pembacaan data file dan melakukan pencarian keyword-keyword tertentu. Gambar 3. 9 – Gambar 3. 17 menunjukkan diagram alir utama dari proses penghitungan nilai metrik pada applikasi yang dibangun ini. Proses-proses ini merupakan proses perhitungan yang dilakukan untuk setiap metriknya.

Tahap membaca file data inputan merupakan tahap awal di mana applikasi memeriksa ketersediaan data file dan melakukan pembacaan terhadap isi file inputan yang memiliki ekstensi java atau txt. Data isi file inputan kemudian disimpan dalam bentuk *string*. Dari data *string* ini kemudian akan dilakukan pencarian berdasarkan keyword untuk melakukan penghitungan nilai kompleksitas metrik.

Gambar 3. 9 menunjukkan diagram alir (*flow chart*) dari proses perhitungan metrik *Cyclomatic Complexity*. Proses yang pertama dilakukan adalah membaca berkas inputan yang diberikan dan menampung data isinya perbarisnya. *String* isi dari baris tersebut kemudian diperiksa apakah ada isinya atau kosong. Apabila isinya kosong, artinya data sudah mencapai baris akhir dan data hasil kalkulasi akan dikelompokkan untuk mengetahui tingkatan kompleksitasnya dan kemudian ditampilkan. Apabila isi dari *string* yang ditampung ada, maka akan dilakukan pemeriksaan kembali, apakah pada string tersebut terdapat salah satu kata kunci yang digunakan untuk mencari adanya nilai *cyclomatic complexity*. Apabila ditemukan salah satu dari kata kunci, maka nilai dari *cyclomatic complexity* akan diperbaharui, sedang apabila tidak ada, maka nilai dari metrik *cyclomatyc complexity* akan tetap. Setelah itu akan dilakukan pembacaan baris berikut hingga ditemukan akhir dari data.



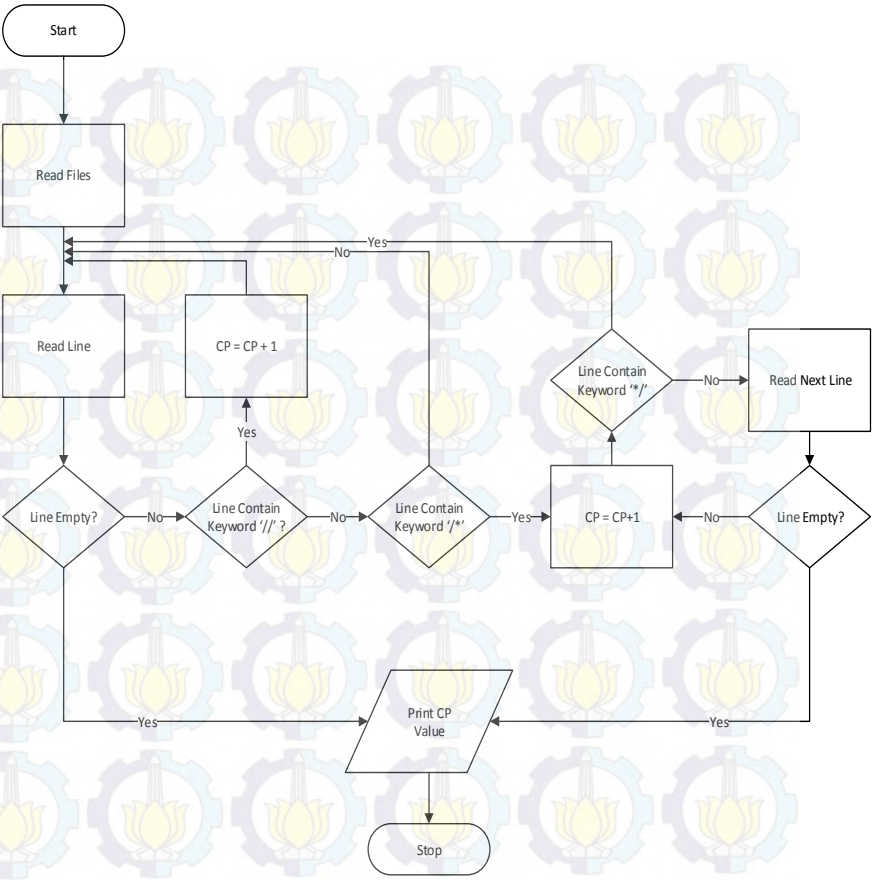
Gambar 3. 9 Diagram Alir Proses Perhitungan *Cyclomatic Complexity*



Gambar 3. 10 Diagram Alir Proses Perhitungan *Depth Of Inheritance*

Gambar 3. 10 menunjukkan diagram alir (*flow chart*) dari proses perhitungan metrik *Depth of Inheritance*. Proses yang pertama dilakukan adalah membaca berkas inputan yang diberikan dan menampung data isinya perbarisnya. *String* isi dari baris tersebut kemudian diperiksa apakah ada isinya atau kosong. Apabila isinya kosong, artinya data sudah mencapai baris akhir dan data hasil kalkulasi akan ditampilkan. Apabila isi dari *string* yang ditampung ada, maka akan dilakukan pemeriksaan kembali, apakah data yang ditampung tersebut berupa *class*. Apabila bukan, maka akan dilanjutkan untuk pembacaan data pada baris berikutnya. Apabila data pada baris tersebut menunjukkan pembentukan *class* yang baru, maka akan diperiksa apakah *class* tersebut memilki *parent class*. Apabila data *parent class* tidak ada, maka akan dilakukan pembacaan baris yang baru, sedang apabila ditemukan adanya *parent class* akan dilanjutkan pemeriksaan kembali apakah *parent class* tersebut juga memiliki *parent class* lagi untuk kemudian nilai dari metriknya ditambahkan. *Looping* ini dilakukan hingga tidak ditemukan adanya *parent class* lagi. Dan kemudian akan dilanjutkan untuk pemeriksaan data baris yang baru lagi.

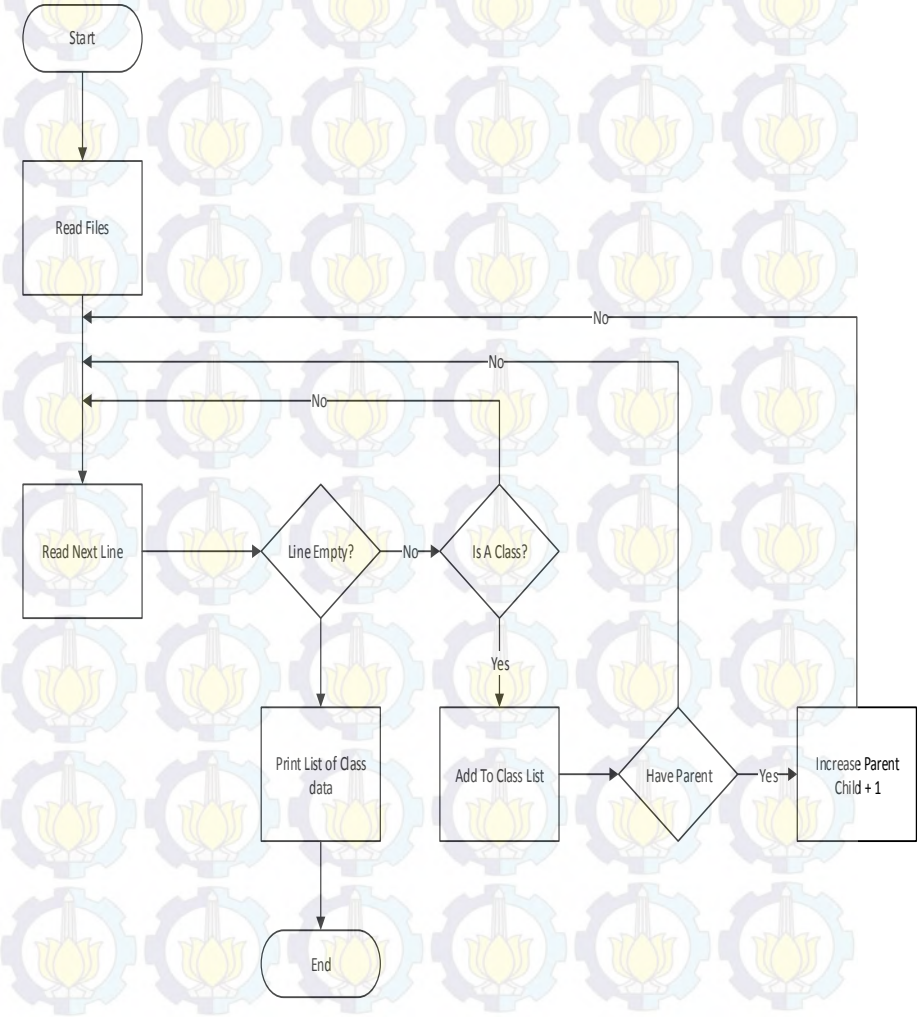
Gambar 3. 11 menunjukkan diagram alir (*flow chart*) dari proses perhitungan metrik *Comment Percentage*. Proses yang pertama dilakukan adalah membaca berkas inputan yang diberikan dan menampung data isinya perbarisnya. *String* isi dari baris tersebut kemudian diperiksa apakah ada isinya atau kosong. Apabila isinya kosong, artinya data sudah mencapai baris akhir dan data hasil kalkulasi akan ditampilkan. Apabila isi dari *string* yang ditampung ada, maka akan dilakukan pemeriksaan kembali, apakah data yang ditampung memiliki kata kunci untuk menandakan adanya permulaan baris *comment*. Apabila data yang ditampung memilikinya, maka jumlah dari metrik akan bertambah satu.



Gambar 3. 11 Diagram Alir Proses Perhitungan *Comment Percentage*

Sedangkan apabila tidak ada maka akan dilakukan pemeriksaan berikut untuk mencari apakah ada kata kunci yang menandakan adanya *comment group*, apabila ada kata kunci yang menandakan awal dari *comment group*, maka jumlah metrik akan bertambah 1 dan kemudian dilakukan *looping* untuk pemeriksaan hingga ditemukan akhir dari *comment group,* sampai akhir dari *comment group* ditemukan maka jumlah metrik akan bertambah

setiap pemeriksaan baris baru. Setelah perhitungan selesai akan dilakukan pemeriksaan kembali hingga ditemukan akhir dari baris dan kemudian data hasil perhitungan akan ditampilkan.

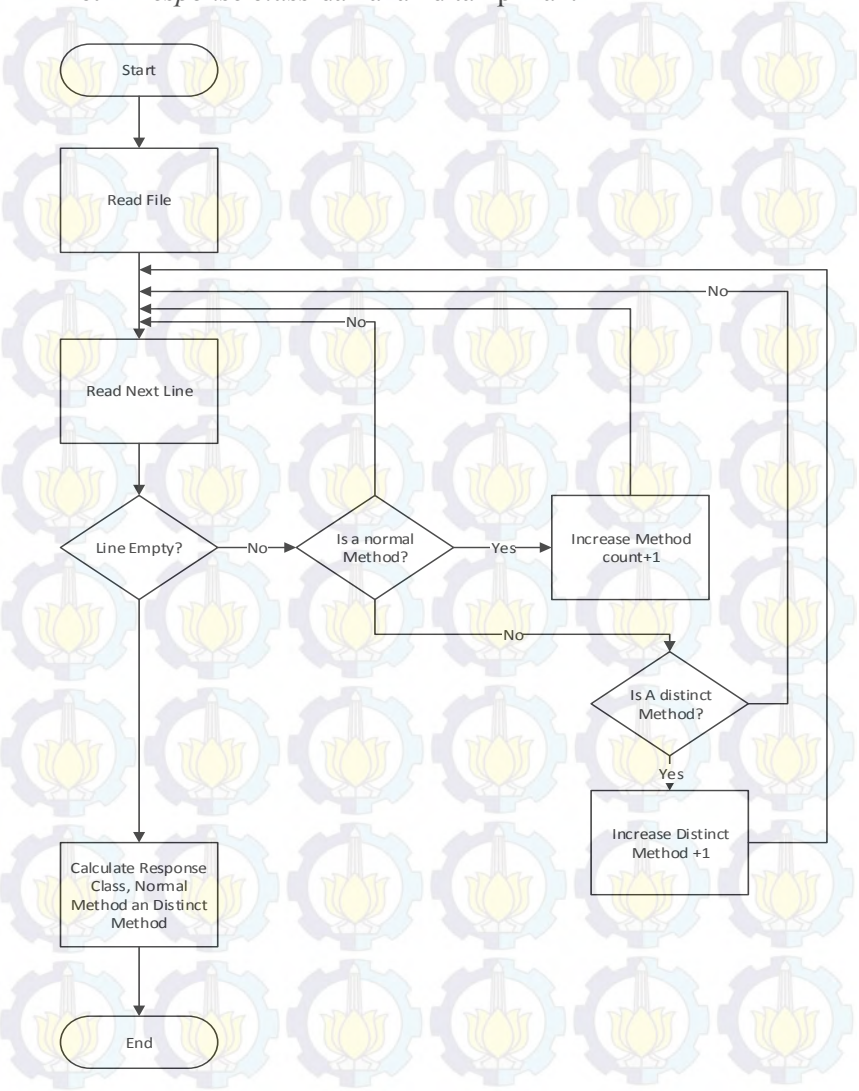


Gambar 3. 12 Diagram Alir Perhitungan *Number of Children*

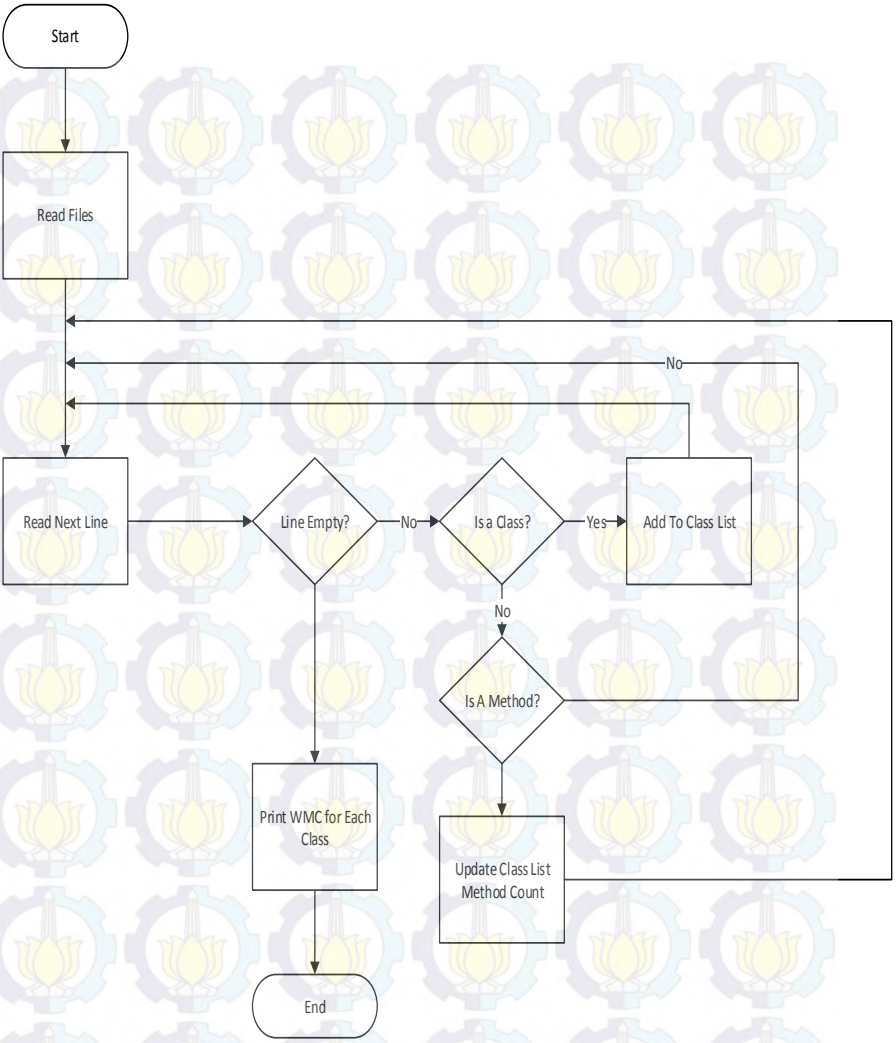
Gambar 3. 12 menunjukkan diagram alir (*flow chart*) dari proses perhitungan metrik *Number of Children*. Proses yang pertama dilakukan adalah membaca berkas inputan yang diberikan dan menampung data isinya perbarisnya. *String* isi dari baris tersebut kemudian diperiksa apakah ada isinya atau kosong. Apabila isinya kosong, artinya data sudah mencapai baris akhir dan data hasil kalkulasi akan ditampilkan. Apabila isi dari *string* yang ditampung ada, maka akan dilakukan pemeriksaan kembali, apakah data yang ditampung tersebut berupa *class*. Apabila bukan, maka akan dilanjutkan untuk pembacaan data pada baris berikutnya. Apabila data pada baris tersebut menunjukkan pembentukan *class* yang baru, maka akan ditampung pada *list class*. Kemudian akan dilakukan pemeriksaan lagi apakah *class* tersebut memilki *parent class*. Apabila data *parent class* tidak ada, maka akan dilakukan pembacaan baris yang baru, sedang apabila ditemukan adanya *parent class* maka akan nilai metrik NOC pada *parent class* tersebut akan bertambah. Kemudian akan dilakukan pemeriksaan pada baris berikutnya hingga ditemukan baris terakhir pada file dan data hasil perhitungan akan ditampilkan.

Gambar 3. 13 menunjukkan diagram alir (*flow chart*) dari proses perhitungan metrik *Response class*. Proses yang pertama dilakukan adalah membaca berkas inputan yang diberikan dan menampung data isinya perbarisnya. *String* isi dari baris tersebut kemudian diperiksa apakah ada isinya atau kosong. Apabila isinya kosong, artinya data sudah mencapai baris akhir dan data hasil kalkulasi akan ditampilkan. Apabila isi dari *string* yang ditampung ada, maka akan dilakukan pemeriksaan kembali, apakah data yang ditampung tersebut berupa *method* atau bukan. Apabila iya maka jumlah normal *method* akan bertambah, sedang apabila bukan akan diperiksa lagi apakah baris tersebut mengandung *distinct method*, apabila iya maka jumlah *distinct method* akan bertambah dan dilakukan pemeriksaan pada baris berikutnya. Setelah baris akhir ditemukan, nilai dari jumlah

method dan *distinct method* akan ditotalkan sebagai nilai dari metrik *response class* dan akan ditampilkan.



Gambar 3. 13 Diagram Alir Perhitungan *Response Class*

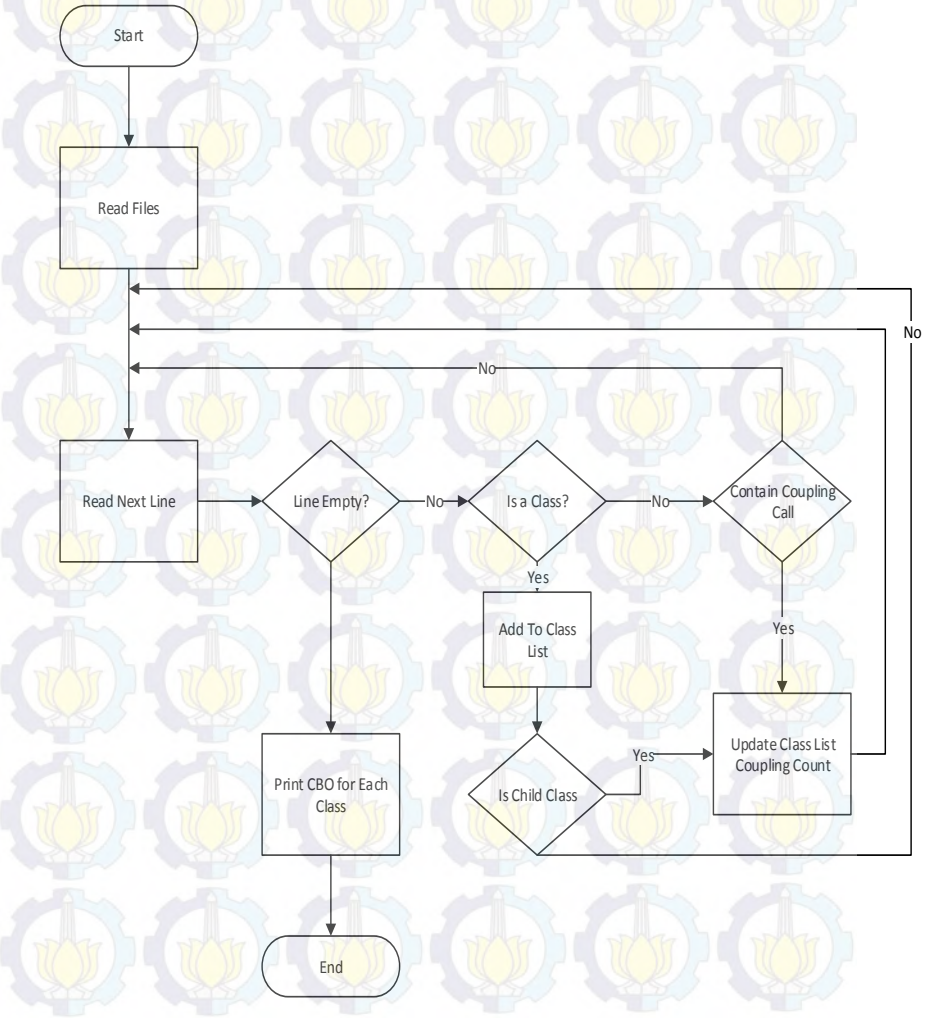


Gambar 3. 14 Diagram Alir Perhitungan *Weighted Method Complexity*

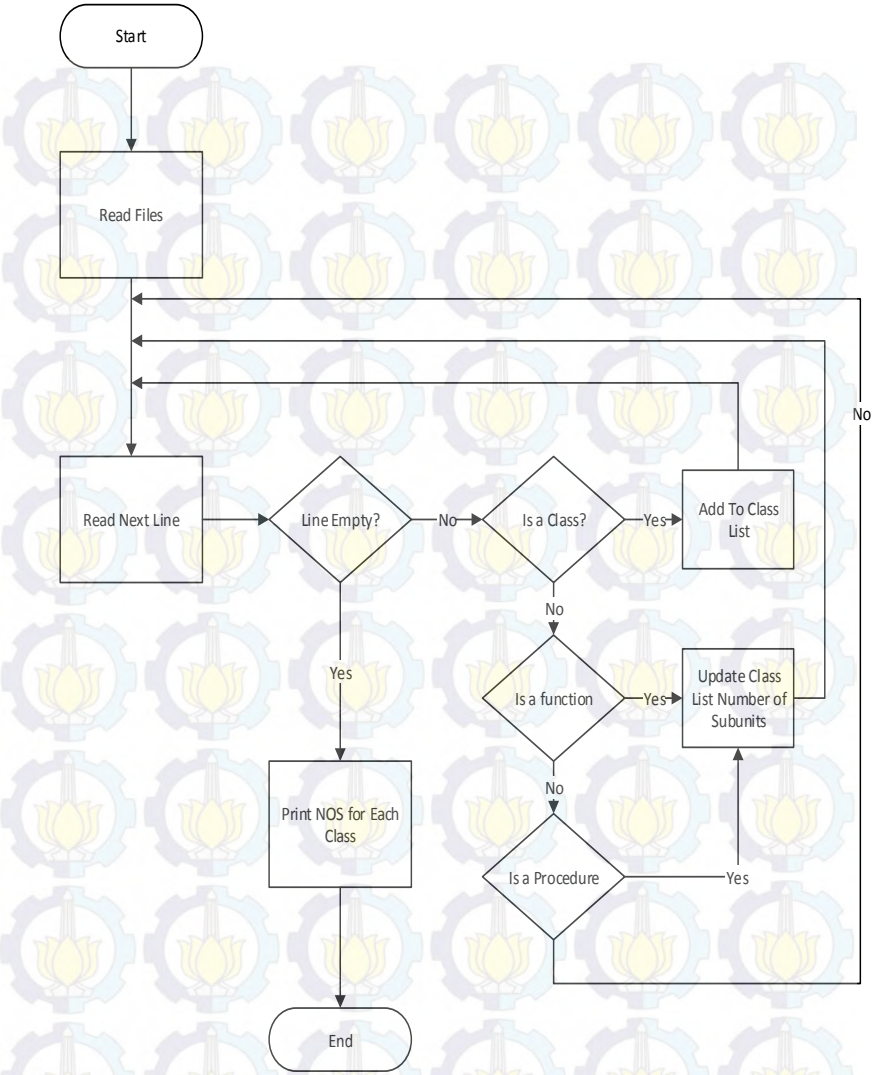
Gambar 3. 14 menunjukkan diagram alir (*flow chart*) dari proses perhitungan metrik *Weighted Method Count.* Proses yang pertama dilakukan adalah membaca berkas inputan yang diberikan dan menampung data isinya perbarisnya. *String* isi dari baris tersebut kemudian diperiksa apakah ada isinya atau kosong. Apabila isinya kosong, artinya data sudah mencapai baris akhir dan data hasil kalkulasi akan ditampilkan. Apabila isi dari *string* yang ditampung ada, maka akan dilakukan pemeriksaan kembali, apakah data yang ditampung merupakan sebuah *class* atau bukan, apabila merupakan *class* maka masukkan data ke dalam *list class*, kemudian baca baris yang baru. Jika data bukan merupakan *class,* maka lakukan pemeriksaan berikut, apakah data merupakan sebuah *method* atau bukan. Apabila data merupakan *method* maka *update* data jumlah *method count* pada *class* yang memilikinya. Setelah itu baca baris berikutnya dan lakukan proses pemeriksaan kembali hingga ditemukan akhir dari baris. Setelah baris akhir ditemukan, nilai metrik *weighted method count* akan ditampilkan.

Gambar 3. 15 menunjukkan diagram alir (*flow chart*) dari proses perhitungan metrik *Coupling Between Object.* Proses yang pertama dilakukan adalah membaca berkas inputan yang diberikan dan menampung data isinya perbarisnya. *String* isi dari baris tersebut kemudian diperiksa apakah ada isinya atau kosong. Apabila isinya kosong, artinya data sudah mencapai baris akhir dan data hasil kalkulasi akan ditampilkan. Apabila isi dari *string* yang ditampung ada, maka akan dilakukan pemeriksaan kembali, apakah terdapat data *class*, yang artinya ada *class* yang baru dan akan di masukkan ke dalam list data *class*. Kemudian *class* tersebut akan diperiksa, apakah dia merupakan *child class* atau tidak. Apabila data tersebut bukan *child class,* maka akan dilanjutkan pembacaan baris yang baru. Sedangkan jika data tersebut merupakan *child class*, maka akan dilakukan pembaharuan total *coupling* yang terdapat pada list *class,* dan akan dilanjutkan untuk membaca baris yang baru. Namun apabila data tersebut bukan *class,* maka akan dilakukan pemeriksaan apakah terdapat pemanggilan *coupling* pada baris tersebut.

Apabila ada maka akan dilakukan pembaharuan jumlah total *coupling* yang terdapat pada list class, sedangkan jika tidak ada maka akan dilakukan pembacaan baris berikutnya.



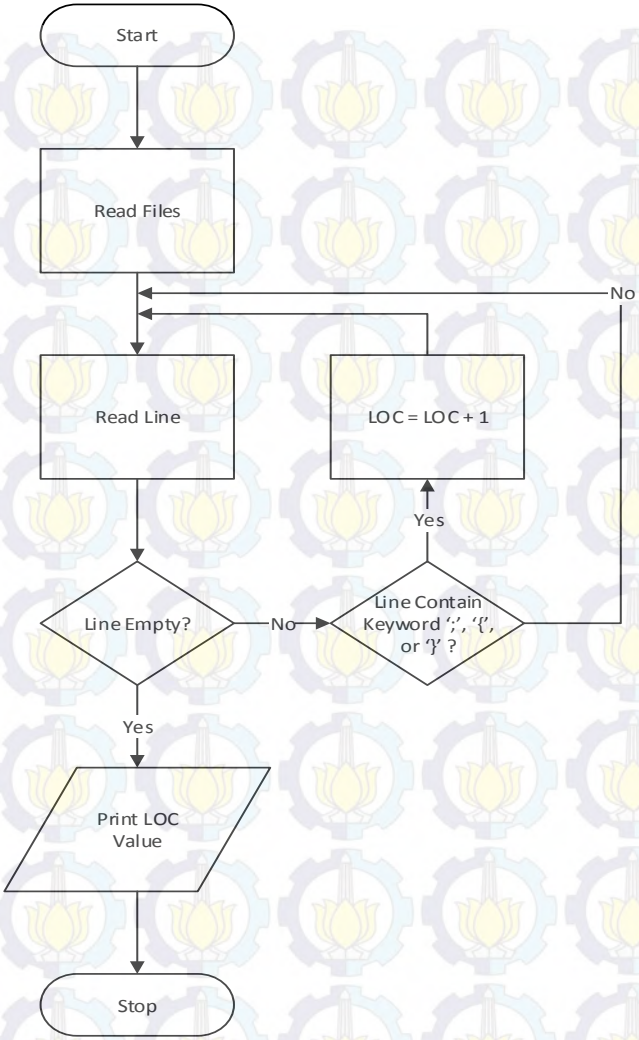
Gambar 3. 15 Diagram Alir Perhitungan *Coupling Between Object*



Gambar 3. 16 Diagram Alir Perhitungan *Number of Subunits*

Gambar 3. 16 menunjukkan diagram alir (*flow chart*) dari proses perhitungan metrik *Number of Subunits*. Proses yang pertama dilakukan adalah membaca berkas inputan yang diberikan dan menampung data isinya perbarisnya. *String* isi dari baris tersebut kemudian diperiksa apakah ada isinya atau kosong. Apabila isinya kosong, artinya data sudah mencapai baris akhir dan data hasil kalkulasi akan ditampilkan. Apabila isi dari *string* yang ditampung ada, maka akan dilakukan pemeriksaan kembali, apakah data yang ditampung tersebut berupa *class*. Apabila bukan, maka akan dilanjutkan untuk pembacaan data pada baris berikutnya. Apabila data pada baris tersebut menunjukkan pembentukan *class* yang baru, maka akan ditampung pada *list class*. Kemudian akan dilakukan pemeriksaan lagi apakah pada baris tersebut terdapat fungsi, jika tidak ada maka akan diperiksa kembali apakah ada prosedur. Apabila fungsi maupun prosedur tidak ditemukan, maka akan dilanjutkan pemeriksaan pada baris yang baru, sedang apabila ada ditemukan fungsi ataupun prosedur, maka jumlah metrik NOS pada kelas tersebut akan bertambah, dan dilanjutkan pemeriksaan baris berikut hingga baris terakhir. Setelah itu data hasil perhitungan metrik akan ditampilkan.

Gambar 3. 17 menunjukkan diagram alir (*flow chart*) dari proses perhitungan metrik *Line of Code*. Proses yang pertama dilakukan adalah membaca berkas inputan yang diberikan dan menampung data isinya perbarisnya. *String* isi dari baris tersebut kemudian diperiksa apakah ada isinya atau kosong. Apabila isinya kosong, artinya data sudah mencapai baris akhir dan data hasil kalkulasi akan ditampilkan. Apabila isi dari *string* yang ditampung ada, maka akan dilakukan pemeriksaan kembali, apakah data yang ditampung tersebut memiliki salah satu dari kata kunci yang menandakan akhir dari satu baris. Apabila salah satu kata kunci terdapat pada data yang ditampung maka nilai dari metrik *Line of Code* akan bertambah. Setelah itu akan dilanjutkan untuk memeriksa baris berikutnya hingga ditemukan baris terakhir.



Gambar 3. 17 Diagram Alir Perhitungan *Line of Code*

3.2.3. Perancangan Antarmuka Pengguna

Bagian ini membahas rancangan tampilan antar muka pada sistem. Pada sistem ini terdapat satu tampilan utama yaitu tampilan yang digunakan oleh pengguna untuk memilih jenis metrik yang diinginkan dan untuk memilih *file* yang ingin diperiksa.

Halaman Tampilan menampilkan hasil deteksi. Halaman ini merupakan tampilan utama yang muncul ketika sistem pertama kali dijalankan. Pada halaman ini terdapat 2 buah tombol, yakni tomboll *Choose file* dan tombol *Calculate*. Pada halaman ini juga terdapat 10 buah *check box* untuk memilih jenis metrik yang ingin dihitung dan sebuah *textbox* yang berisi data *file* yang dipilih pengguna untuk diperiksa. Pengguna dapat memilih *file* terlebih dahulu dengan menekan tombol *Choose File,* dan kemudian alamat *file* akan tampil pada *textbox* yang tersedia. Setelah itu pengguna dapat memilih jenis metrik yang ingin dilakukan perhitungan dan kemudian menekan tombol *Calculate.* Hasil dari perhitungan kemudian akan disimpan dalam bentuk *file .pdf* yang akan ditampilkan setelah tersimpan. Pada *file* ini akan ditampilkan hasil perhitungan berupa teks dan diagram batang yang menunjukkan nilai dari masing-masing metrik.

Tabel 3. 5 Spesifikasi Atribut Antar Muka Tampilan Utama

No	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan	Jenis Masukan / Keluaran
1	Tombol *Choose file*	JButton	Memilih *file* yang akan digunakan sebagai data masukan	*Action*/Alamat *file*
2	Teks Box Alamat	JTextField	Menampilkan alamat *file* yang akan dijadikan sebagai data masukan	*Action*/Alamat *file*

Tabel 3.5 Spesifikasi Atribut Antar Muka Tampilan Utama

No	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan	Jenis Masukan / Keluaran
3	*Check box* jenis metrik	JCheckBox	Menentukan apakah metrik tersebut akan masuk dalam proses kalkulasi atau tidak	*Action/Boolean*
4	Tombol *Calculate*	JButton	Melakukan proses perhitungan pada data masukan untuk setiap metrik yang telah dipilih.	*Action/Document*

BAB IV
IMPLEMENTASI

Bab ini membahas tentang implementasi dari perancangan sistem. Bab ini berisi proses implementasi dari setiap kelas pada semua modul. Bahasa pemrograman yang digunakan adalah bahasa pemrograman Java.

4.1 Implementasi Lapisan Antarmuka

Lapisan antarmuka merupakan lapisan yang bertugas mengatur tampilan sistem. Pada bagian ini akan dijelaskan implementasi dari rancangan lapisan antarmuka.

4.1.1 Kelas MainForm

Kelas ini merupakan kelas antar muka yang digunakan sebagai kelas utama dan kelas yang menampung *view* pada applikasi. Kelas ini merupakan turunan dari kelas abstrak *JFrame*. Pada kelas ini terdapat fungsi-fungsi yang digunakan sebagai kontrol dari setiap komponen *Java Swing* yang digunakan pada applikasi ini.

Fungsi *btnChooseFileActionPerformed* merupakan fungsi yang digunakan untuk mendengarkan aksi yang diberikan pada tombol *Choose File*. Fungsi ini berguna untuk membuka jendela baru bagi pengguna untuk memilih *file* yang ingin digunakan sebagai data masukan. Fungsi ini dapat dilihat pada Kode 4. 1. Pada fungsi ini dilakukan pembentukan komponen *JFileChooser* baru, dan apabila *file* yang diinginkan telah dipilih, maka alamat *file* tersebut akan disimpan pada variabel *myFilePath.*

Fungsi *btnCalculateActionPerformed* merupakan fungsi yang digunakan untuk mendengarkan aksi yang diberikan pada tombol *Calculate.* Fungsi ini berguna untuk memanggil proses kalkulasi metrik yang telah dipilih dan kemudian menampung data hasilnya untuk kemudian digunakan sebagai isi dari berkas pdf hasil akhir.

Fungsi *generateBarChart* merupakan fungsi yang digunakan untuk membentuk diagram batang berdasarkan hasil kalkulasi dari setiap metrik yang telah dipilih untuk dikalkulasi. Diagram-diagram batang yang telah dibentuk kemudian ditampung pada variabel *allCharts* yang nantinya akan ditampilkan pada berkas pdf hasil akhir.

```
private void btnChooseFileActionPerformed
(java.awt.event.ActionEvent evt) {
  JFileChooser fileChooser = new JFileChooser();
  fileChooser.addChoosableFileFilter(new
FileNameExtensionFilter("TEXT FILE",".txt", "txt"));
  fileChooser.addChoosableFileFilter(new
FileNameExtensionFilter("JAVA FILE","java", "JAVA",
"Java"));
  int returnValue = fileChooser.showOpenDialog(null);
  if (returnValue == JFileChooser.APPROVE_OPTION) {
    File selectedFile = fileChooser.getSelectedFile();
    myFilePath = selectedFile.getAbsolutePath();
    txtPath.setText(myFilePath);
  }
}
private void btnCalculateActionPerformed
(java.awt.event.ActionEvent evt) {
  if (myFilePath==null) {
    JOptionPane.showConfirmDialog(this, "Please choose
file first", "File Not Choosen",
JOptionPane.ERROR_MESSAGE);
  }else if (!myFilePath.isEmpty() ||
!myFilePath.equals("") || myFilePath!=null) {
    try {
      allCharts.removeAll(allCharts);
      FileReader fr;
      BufferedReader br;
      List<PDFData> allData = new ArrayList<>();
      PDFData pdfData;
      if (chkDIT.isSelected()) {
        fr = new FileReader(myFilePath);
        br = new BufferedReader(fr);
             DepthOfInheritance dit = new

DepthOfInheritance();
```

```
        StringBuilder myDIT = dit.getClassDIT(br);
        String type = "Depth Of Inheritance";
        pdfData = new PDFData();
        pdfData.setData(myDIT);
        pdfData.setType(type);
        allData.add(pdfData);
        generateBarChart(dit.getInheritanceList(),
"DOI");
      }
      if (chkCC.isSelected()) {
        fr = new FileReader(myFilePath);
        br = new BufferedReader(fr);
        CyclomaticComplexity cc = new
CyclomaticComplexity();
        StringBuilder myCC = cc.getCComplexity(br);
        String type = "Cyclomatic Complexity";
        pdfData = new PDFData();
        pdfData.setData(myCC);
        pdfData.setType(type);
        allData.add(pdfData);
      }
      if (chkNOC.isSelected()) {
        fr = new FileReader(myFilePath);
        br = new BufferedReader(fr);
        NumberOfChildren noc = new NumberOfChildren();
        StringBuilder myNOC = noc.getClassNOC(br);
        String type = "Number Of Children";
        pdfData = new PDFData();
        pdfData.setData(myNOC);
        pdfData.setType(type);
        allData.add(pdfData);
        generateBarChart(noc.getChildrenList(),
"NOC");
      }
      if (chkRFC.isSelected()) {
        fr = new FileReader(myFilePath);
        br = new BufferedReader(fr);
        ResponseClass rfc = new ResponseClass();
        StringBuilder myRFC =
rfc.getResponseForClass(br);
```

```
        String type = "Response For Class";
        pdfData = new PDFData();

pdfData.setData(myRFC);
        pdfData.setType(type);
        allData.add(pdfData);
      }
      if (chkWMC.isSelected()) {
        fr = new FileReader(myFilePath);
        br = new BufferedReader(fr);
        WeightedMethod wm = new WeightedMethod();
        StringBuilder myWMC =
wm.getWeightedMethod(br);
        String type = "Weighted Method for Class";
        pdfData = new PDFData();
        pdfData.setData(myWMC);
        pdfData.setType(type);
        allData.add(pdfData);
        generateBarChart(wm.getMethodList(), "WMC");
      }
      if (chkCP.isSelected()) {
        fr = new FileReader(myFilePath);
        br = new BufferedReader(fr);
        CommentPercentage cp = new
CommentPercentage();
        StringBuilder myCp =
cp.getCommentPercentage(br);
        String type = "Comment Percentage";
        pdfData = new PDFData();
        pdfData.setData(myCp);
        pdfData.setType(type);
        allData.add(pdfData);
      }
      if (chkFZ.isSelected()) {
        MyFileSize fz = new MyFileSize();
        StringBuilder myFZ = fz.check(myFilePath);
        String type = "File Size";
        pdfData = new PDFData();
        pdfData.setData(myFZ);
        pdfData.setType(type);
        allData.add(pdfData);
      }
      if (chkCBO.isSelected()) {
```

```
        CouplingBetweenObject cbo = new
CouplingBetweenObject();
        StringBuilder myCBO =
cbo.getClassCBO(myFilePath);
        String type = "Coupling Between Objects";
        pdfData = new PDFData();
        pdfData.setData(myCBO);
        pdfData.setType(type);
        allData.add(pdfData);
        generateBarChart(cbo.getCouplingList(),
"CBO");
      }
      if (chkNOS.isSelected()) {
        fr = new FileReader(myFilePath);
        br = new BufferedReader(fr);
        NumberOfSubunits ns = new NumberOfSubunits();
        StringBuilder myNOS = ns.getSubUnits(br);
        String type = "Number of Subunits";
        pdfData = new PDFData();
        pdfData.setData(myNOS);
        pdfData.setType(type);
        allData.add(pdfData);
        generateBarChart(ns.getSubunitList(), "NOS");
      }
      if (chkLOC.isSelected()) {
        fr = new FileReader(myFilePath);
        br = new BufferedReader(fr);
        LineOfCode loc = new LineOfCode();
        StringBuilder myLoc = loc.getLineOfCode(br);
        String type = "Line Of Code";
        pdfData = new PDFData();
        pdfData.setData(myLoc);
        pdfData.setType(type);
        allData.add(pdfData);
      }
      PDFCreator pc = new PDFCreator();
      pc.createPdf(allData, allCharts);
    } catch (FileNotFoundException | HeadlessException
e) {
```

```
System.out.println(e);
    }
  }
}
public void generateBarChart(List<ClassData> listObj,
String type) {
  DefaultCategoryDataset dataSet = new
DefaultCategoryDataset();
  if (type.equals("NOC")) {
    for (ClassData noc : listObj) {
    dataSet.setValue(noc.getValue(), "Number Of
Children", noc.getClassName());
  }
  JFreeChart chart = ChartFactory.createBarChart(
    "Number Of Children", "Class Name", "NOC Value",
    dataSet, PlotOrientation.VERTICAL, false, true,
false);
  allCharts.add(chart);
  }else if (type.equals("DOI")) {
    for (ClassData co : listObj) {
    dataSet.setValue(co.getValue(), "Depth Of
Inheritance", co.getClassName());
  }
  JFreeChart chart = ChartFactory.createBarChart(
    "Depth Of Inheritance", "Class Name", "DOI Value",
    dataSet, PlotOrientation.VERTICAL, false, true,
false);
  allCharts.add(chart);
  }else if (type.equals("CBO")) {
    for (ClassData co : listObj) {
    dataSet.setValue(co.getValue(), "Coupling Between
Objects", co.getClassName());
  }
  JFreeChart chart = ChartFactory.createBarChart(
    "Coupling Between Objects", "Class Name", "CBO
Value",
    dataSet, PlotOrientation.VERTICAL, false, true,
false);
  allCharts.add(chart);
  }else if (type.equals("WMC")) {
            for (ClassData co : listObj) {
```

```
      dataSet.setValue(co.getValue(), "Weighted Method
for Class", co.getClassName());
    }
    JFreeChart chart = ChartFactory.createBarChart(
      "Weighted Method for Class", "Class Name", "WMC
Value",
      dataSet, PlotOrientation.VERTICAL, false, true,
false);
    allCharts.add(chart);
  }else if (type.equals("NOS")) {
    for (ClassData co : listObj) {
      dataSet.setValue(co.getValue(), "Number of
Subunits", co.getClassName());
    }

    JFreeChart chart = ChartFactory.createBarChart(
      "Number of Subunits", "Class Name", "NOS Value",
      dataSet, PlotOrientation.VERTICAL, false, true,
false);
    allCharts.add(chart);
  }
}
```

Kode 4. 1 Fungsi-fungsi yang terdapat pada kelas *MainForm*

4.2 Implementasi Lapisan Kontrol

Lapisan kontrol merupakan lapisan yang bertanggung jawab dengan tingkah laku sistem. Lapisan ini bertugas menghubungkan lapisan entity dengan lapisan antarmuka. Pada lapisan ini dilakukan pemanggilan nilai kalkulasi dari setiap metrik yang terdapat pada lapisan entity.

4.2.1 Kelas CommentPercentageManager

Kelas ini merupakan kelas *controller* dari metrik *comment percentage.* Pada kelas ini terdapat sebuah fungsi yang berguna untuk melakukan pemanggilan hasil kalkulasi dari metrik *comment percentage,* yaitu fungsi *getCommentPercentage.*

```
public StringBuilder getCommentPercentage
(BufferedReader br){
   CommentPercentage cp = new CommentPercentage();
   return cp.getCommentPercentage(br);
}
```

Kode 4. 2 Fungsi-fungsi yang terdapat pada kelas CommentPercentageManager

4.2.2 Kelas CouplingBetweenObjectManager

Kelas ini merupakan kelas *controller* dari metrik *coupling between object.* Pada kelas ini terdapat dua buah fungsi, yaitu fungsi *getClassCBO* yang berguna untuk melakukan pemanggilan hasil kalkulasi dari metrik *class between object,* dan fungsi *getCouplingList* yang berfungsi untuk memanggil data hasil kalkulasi setiap kelas untuk nantinya ditampilkan pada *bar chart.*

```
public List<ClassData> getCouplingList() {
   return cbo.getCouplingList();
}
public StringBuilder getClassCBO(String myFilePath)
throws FileNotFoundException{
   return cbo.getClassCBO(myFilePath);
}
```

Kode 4. 3 Fungsi-fungsi yang terdapat pada kelas *ClassBetweenObjectManager*

4.2.3 Kelas CyclomaticComplexityManager

Kelas ini merupakan kelas *controller* dari metrik *cyclomatic complexity.* Pada kelas ini terdapat sebuah fungsi yang berguna untuk melakukan pemanggilan hasil kalkulasi dari metrik *cyclomatic complexity,* yaitu fungsi *getCComplexity.*

```
public StringBuilder getCComplexity(BufferedReader
br){
  CyclomaticComplexity cc = new
CyclomaticComplexity();
  return cc.getCComplexity(br);
}
```

Kode 4. 4 Fungsi-fungsi yang terdapat pada kelas *CyclomaticComplexityManager*

4.2.4 Kelas DepthOfInheritanceManager

Kelas ini merupakan kelas *controller* dari metrik *depth of inheritance.* Pada kelas ini terdapat dua buah fungsi, yaitu fungsi *getClassDIT* yang berguna untuk melakukan pemanggilan hasil kalkulasi dari metrik *depth of inheritance,* dan fungsi *getInheritanceList* yang berfungsi untuk memanggil data hasil kalkulasi setiap kelas untuk nantinya ditampilkan pada *bar chart.*

```
public List<ClassData> getInheritanceList() {
   return dit.getInheritanceList();
}
public StringBuilder getClassDIT(BufferedReader br) {
   return dit.getClassDIT(br);
}
```

Kode 4. 5 Fungsi-fungsi yang terdapat pada kelas *DepthOfInheritanceManager*

4.2.5 Kelas LineOfCodeManager

Kelas ini merupakan kelas *controller* dari metrik *line of code.* Pada kelas ini terdapat sebuah fungsi yang berguna untuk melakukan pemanggilan hasil kalkulasi dari metrik *line of code,* yaitu fungsi *getLineOfCode.*

```
public StringBuilder getLineOfCode(BufferedReader br){
   LineOfCode loc = new LineOfCode();
   return loc.getLineOfCode(br);
}
```

Kode 4. 6 Fungsi-fungsi yang terdapat pada kelas *LineOfCodeManager*

4.2.6 Kelas MyFileSizeManager

Kelas ini merupakan kelas *controller* dari metrik *file size.* Pada kelas ini terdapat sebuah fungsi yang berguna untuk melakukan pemanggilan hasil kalkulasi dari metrik *file size,* yaitu fungsi *check.*

```
public StringBuilder check(String fileName){
   MyFileSize fs = new MyFileSize();
   return fs.check(fileName);
}
```

Kode 4. 7 Fungsi-fungsi yang terdapat pada kelas *MyFileSizeManager*

4.2.7 Kelas NumberOfChildrenManager

Kelas ini merupakan kelas *controller* dari metrik *number of children.* Pada kelas ini terdapat dua buah fungsi, yaitu fungsi *getClassNOC* yang berguna untuk melakukan pemanggilan hasil kalkulasi dari metrik *number of children,* dan fungsi *getChildrenList* yang berfungsi untuk memanggil data hasil kalkulasi setiap kelas untuk nantinya ditampilkan pada *bar chart.*

```
public List<ClassData> getChildrenList() {
   return noc.getChildrenList();
}
public StringBuilder getClassNOC(BufferedReader br) {
   return noc.getClassNOC(br);
}
```

Kode 4. 8 Fungsi-fungsi yang terdapat pada kelas *NumberOfChildrenManager*

4.2.8 Kelas NumberOfSubunitsManager

Kelas ini merupakan kelas *controller* dari metrik *number of subunits.* Pada kelas ini terdapat dua buah fungsi, yaitu fungsi *getSubUnits* yang berguna untuk melakukan pemanggilan hasil kalkulasi dari metrik *number of subunits,* dan fungsi *getSubunitList* yang berfungsi untuk memanggil data hasil kalkulasi setiap kelas untuk nantinya ditampilkan pada *bar chart.*

```
public List<ClassData> getSubunitList() {
   return nos.getSubunitList();
}
public StringBuilder getSubUnits(BufferedReader br) {
   return nos.getSubUnits(br);
}
```

Kode 4. 9 Fungsi-fungsi yang terdapat pada kelas *NumberOfSubunitsManager*

4.2.9 Kelas PDFCreator

Kelas ini merupakan kelas yang berfungsi untuk membuat berkas pdf dari hasil akhir. Kelas ini bertugas untuk membuat berkas pdf baru dan menuliskan hasil kalkulasi dan dan hasil dari *bar chart* setelah kalkulasi ke dalam berkas pdf. Pada kelas ini terdapat 4 buah *method*, yaitu *createPdf, addMetaData, addContent,* dan *addEmptyLine.* Method *createPdf* berguna untuk menciptakan sebuah berkas pdf baru sebagai hasil akhir. Method *addMetaData* berguna untuk mengisi data *properties* dari berkas pdf. Method *addContent* berguna untuk memasukkan data hasil sebagai isi dari berkas pdf. Sedangkan method *addEmptyLine* berguna untuk memberikan spasi baris kosong pada data yang dimasukkan.

```
public void createPdf(List<PDFData> myData,
List<JFreeChart> charts) {
  try {
    File jarPath=new
File(PDFCreator.class.getProtectionDomain().getCodeSou
rce().getLocation().getPath());
    String FILE = "";
    Date date= new Date();
    long time = date.getTime();
    Timestamp ts = new Timestamp(time);
    FILE = jarPath+"/Hasil_"+ts.toString()+".pdf";
    Document document = new Document();
    writer = PdfWriter.getInstance(document, new
FileOutputStream(FILE));
    document.open();
```

```
    addMetaData(document);
     addContent(document, myData, charts);
     document.close();
     Desktop desktop = Desktop.getDesktop();
     File file = new File(FILE);
     desktop.open(file);
   } catch (FileNotFoundException | DocumentException
e) {
     System.out.println(e);
   } catch (IOException ex) {

Logger.getLogger(PDFCreator.class.getName()).log(Level
.SEVERE, null, ex);
   }
}
private static void addMetaData(Document document) {
   document.addTitle("Metryc Analyzer");
   document.addSubject("Using iText");
   document.addKeywords("Java, PDF, iText");
   document.addAuthor("Wati Marpaung");
   document.addCreator("Wati Marpaung");
}
private static void addContent(Document document,
List<PDFData> myData, List<JFreeChart> charts) throws
DocumentException {
   Paragraph myTitle = new Paragraph("Metryc Analyzer",
catFont);
   addEmptyLine(myTitle, 1);
   document.add(myTitle)
   for (PDFData data : myData) {
     Paragraph subSect = new Paragraph(data.getType(),
subFont);
     document.add(subSect);
     Paragraph value = new
Paragraph(data.getData().toString(), smallBold);
     document.add(value);
     addEmptyLine(value, 1);
     for (JFreeChart chart : charts) {
                                  if
 (chart.getTitle().getText().equals(data.getType())) {
```

```
        PdfContentByte contentByte =
writer.getDirectContent();
        PdfTemplate template =
contentByte.createTemplate(450, 550);
        Graphics2D graphics2d =
template.createGraphics(450, 550, new
DefaultFontMapper());
        Rectangle2D rectangle2d = new
Rectangle2D.Double(0, 0, 500, 400);
        chart.draw(graphics2d, rectangle2d, null);
        graphics2d.dispose();
        contentByte.addTemplate(template, 0, 0);
        document.newPage();
      }
    }
  }
}
private static void addEmptyLine(Paragraph paragraph,
int number) {
  for (int i = 0; i < number; i++) {
    paragraph.add(new Paragraph(" "));
  }
}
```

Kode 4. 10 Fungsi-fungsi yang terdapat pada kelas *PDFCreator*

4.2.10 Kelas ResponseClassManager

Kelas ini merupakan kelas *controller* dari metrik *response for class.* Pada kelas ini terdapat dua buah fungsi, yaitu fungsi *getResponseForClass* yang berguna untuk melakukan pemanggilan hasil kalkulasi dari metrik *response for class,* dan fungsi *getResponseList* yang berfungsi untuk memanggil data hasil kalkulasi setiap kelas untuk nantinya ditampilkan pada *bar chart.*

```
public List<ClassData> getResponseList() {
   return rc.getResponseList();
}
public StringBuilder
getResponseForClass(BufferedReader br) {
   return rc.getResponseForClass(br);
}
```

Kode 4. 11 Fungsi-fungsi yang terdapat pada kelas *ResponseClassManager*

4.2.11 Kelas WeightedMethodManager

Kelas ini merupakan kelas *controller* dari metrik *weighted method complexity.* Pada kelas ini terdapat dua buah fungsi, yaitu fungsi *getWeightedMethod* yang berguna untuk melakukan pemanggilan hasil kalkulasi dari metrik *weighted method complexity,* dan fungsi *getMethodList* yang berfungsi untuk memanggil data hasil kalkulasi setiap kelas untuk nantinya ditampilkan pada *bar chart.*

```
public List<ClassData> getMethodList() {
   return wmc.getMethodList();
}
public StringBuilder getWeightedMethod(BufferedReader
br) {
   return wmc.getWeightedMethod(br);
}
```

Kode 4. 12 Fungsi-fungsi yang terdapat pada kelas *WeightedMethodManager*

4.3 Implementasi Lapisan Entity

Lapisan *entity* merupakan lapisan yang bertanggung jawab dengan tingkah laku sistem. Lapisan ini bertugas melakukan perhitungan dari setiap metrik yang terdapat pada kakas bantu ini.

4.3.1 Kelas Comment Percentage

Kelas ini merupakan kelas yang berfungsi untuk melakukan perhitungan metrik *comment percentage*. Pada kelas ini terdapat dua buah fungsi, yaitu fungsi *check* dan fungsi

getCommentPercentage. Fungsi *check* berguna untuk melakukan proses perhitungan nilai metrik *comment percentage* dari inputan yang diberikan, sedangkan fungsi *getCommentPercentage* berguna untuk melakukan pengelompokan mengenai jenis dari *comment percentage* berkas inputan tersebut.

```
public int check(BufferedReader br) {
  String line="";
  int count=0;
  try {
    while((line=br.readLine())!=null){
      if(line.startsWith("//")) {count++;}
      if(line.startsWith("/*")) {
        count++;
        while(!(line=br.readLine()).endsWith("'*\'"))
{
          count++;
          break;
        }
      }
    }
    br.close();
  }catch (FileNotFoundException e) {
    e.printStackTrace();
  }catch (IOException e) {
    e.printStackTrace();
  }
  return count;
}

public StringBuilder getCommentPercentage
(BufferedReader br){
  StringBuilder ret = new StringBuilder();
  int cpValue = check(br);
  ret.append("The Comment Percentage is : ")
    .append(cpValue);
  ret.append("\n");
        if (cpValue>=150) {
```

```
    ret.append("This program have a Big size");
  }else if (cpValue>=120) {
    ret.append("This program have a Medium-Big size");
  }else if (cpValue>=90) {
    ret.append("This program have a Small-Medium
size");
  }else if (cpValue>=60) {
    ret.append("This program have a Small size");
  }
  ret.append("\n");
  return ret;
}
```

Kode 4. 13 Fungsi-fungsi yang terdapat pada kelas *CommentPercentage*

4.3.2 Kelas CouplingBetweenObject

Kelas ini merupakan kelas yang berfungsi untuk melakukan perhitungan metrik *coupling between object*. Pada kelas ini terdapat tiga buah fungsi, yaitu fungsi *check, getListClasses* dan fungsi *getClassCBO*. Fungsi *check* berguna untuk melakukan proses perhitungan nilai metrik *coupling between object* dari inputan yang diberikan. Fungsi *getListClasses* berguna untuk menampung data setiap class yang ada pada berkas inputan yang diberikan yaitu data nama dan nilai dari *CBO*-nya. Fungsi *getClassCBO* berguna untuk melakukan pengelompokan mengenai jenis dari *coupling between object* berkas inputan tersebut.

```
public List<ClassData> check(String myFilePath) throws
FileNotFoundException{
  FileReader fr;
  FileReader newFr;
  BufferedReader br;
  BufferedReader newBr;

  fr = new FileReader(myFilePath);
  newFr = new FileReader(myFilePath);

  br = new BufferedReader(fr);
```

```
   newBr = new BufferedReader(newFr);
   ClassData thisCo;
   getListClasses(newBr);
   String line = null;
   try {
     line = br.readLine();
     thisCo = new ClassData();
     int thisNumber = 0;
     String thisClass = "";
     while (line != null) {
       String[] arrayString = line.split(" ");
       for(int i=0; i<arrayString.length; i++){
         if(arrayString[i].equals("class")){
           thisClass = arrayString[i+1];
           if (arrayString.length>(i+2)) {
             if(arrayString[i+2]!=null){
               for (int j = 0; j < couplingList.size();
j++) {
                 ClassData co = couplingList.get(j);
                 if
(thisClass.equals(co.getClassName())) {
                   thisCo = co;
                   thisNumber = j;
                 }
               }
               for (int j = 0; j < couplingList.size();
j++) {
                 ClassData co = couplingList.get(j);
                 if
(arrayString[i+3].equals(co.getClassName())) {
                   co.setValue(co.getValue()+1);

thisCo.setValue(thisCo.getValue()+1);
                   couplingList.set(j, co);
                   couplingList.set(thisNumber,
thisCo);
                 }
               }
             }
```

```
          }
        }else{
          if (!line.contains("class")) {
            if (!thisClass.equals("")) {
              for (int j = 0; j < couplingList.size();
j++) {
                ClassData co = couplingList.get(j);
                if
(thisClass.equals(co.getClassName())) {
                  thisCo = co;
                  thisNumber = j;
                }
              }
              for (int j = 0; j < couplingList.size();
j++) {
                ClassData co = couplingList.get(j);
                if
(arrayString[i].equals(co.getClassName())) {
                  if (i>0) {
                    if (!arrayString[i-
1].equals("new")) {
                      co.setValue(co.getValue()+1);
                      couplingList.set(j, co);
                      couplingList.set(thisNumber,
thisCo);
                    }
                  }else{
                    co.setValue(co.getValue()+1);
                    couplingList.set(j, co);
                    couplingList.set(thisNumber,
thisCo);
                  }
                }
              }
            }
          }
        }
      line = br.readLine();
                              }
```

```
    } catch (IOException e) {
      e.printStackTrace();
    }
    return couplingList;
}

private void getListClasses(BufferedReader br) {
    String line = null;
    try {
      line = br.readLine();
      while (line != null) {
        String[] arrayString = line.split(" ");
        for(int i=0; i<arrayString.length; i++){
          if(arrayString[i].equals("class")){
            ClassData co = new ClassData();
            co.setClassName(arrayString[i+1]);
            co.setValue(0);
            couplingList.add(co);
          }
        }
        line = br.readLine();
      }
    } catch (IOException ex) {

Logger.getLogger(CouplingBetweenObject.class.getName()
).log(Level.SEVERE, null, ex);
    }
    System.err.println("Test");
}

public StringBuilder getClassCBO(String myFilePath)
throws FileNotFoundException {
    List<ClassData> couplingObjects = check(myFilePath);
    StringBuilder ret = new StringBuilder();
    int x = 0;
    ret.append("The Coupling Between Objects value : ");
    ret.append("\n");
    for(Object objCBO : couplingObjects){
```

```
    ClassData cbo = (ClassData)objCBO;
    ret.append("For Class ")
      .append(cbo.getClassName())
      .append(" = ")
      .append(cbo.getValue());
    ret.append("\n");
    x = x+cbo.getValue();
  }
  if (x>=30) {
    ret.append("This program have a High Coupling, For
the total CBO = ")
      .append(x);
  }else if(x>20){
    ret.append("This program have a Medium-High
Coupling, For the total CBO = ")
    .append(x);
  }else if(x>10){
    ret.append("This program have a Low-Medium
Coupling, For the total CBO = ")
    .append(x);
  }else{
    ret.append("This program have a Low Coupling, For
the total CBO = ")
    .append(x);
  }
  return ret;
}
```

Kode 4. 14 Fungsi-fungsi yang terdapat pada kelas *CouplingBetweenObject*

4.3.3 Kelas CyclomaticComplexity

Kelas ini merupakan kelas yang berfungsi untuk melakukan perhitungan metrik *cyclomatic complexity*. Pada kelas ini terdapat dua buah fungsi, yaitu fungsi *check* dan fungsi *getCComplexity*. Fungsi *check* berguna untuk melakukan proses perhitungan nilai metrik *cyclomatic complexity* dari inputan yang diberikan. Fungsi *getCComplexity* berguna untuk melakukan pengelompokan mengenai jenis dari *cyclomatic complexity* berkas inputan tersebut.

```
public int check(BufferedReader br) {
  int complexity = 1;
  String[] keywords = {"if", "else", "while", "case",
"for", "switch", "do", "continue", "break", "&&",
    "||", "?", ":", "catch", "finally", "throw",
"throws", "default", "return", "else if", "or",
"and"};
  String words = "";
  String line = null;
  try {
    line = br.readLine();
    while (line != null) {
      StringTokenizer stTokenizer = new
StringTokenizer(line);
      while (stTokenizer.hasMoreTokens()) {
        words = stTokenizer.nextToken();
        for (int i = 0; i < keywords.length; i++) {
          if (words.equals("//")) {break;}
          else {
            if (keywords[i].equals(words)) {
              complexity++;
            }
          }
        }
      }
      line = br.readLine();
    }
  } catch (IOException e) {
    e.printStackTrace();
  }
  return (complexity);
}
public StringBuilder getCComplexity(BufferedReader
br){
  StringBuilder ret = new StringBuilder();
  int ccValue = check(br);
ret.append("The Cyclomatic Complexity is : ");
...
```

```
  ret.append(ccValue);
  ret.append("\n");
  ret.append("Result : ");
  if (ccValue >= 50) {
    ret.append("This program have a High Complexity
");
  } else if (ccValue >= 21) {
    ret.append("This program have a Medium-High
Complexity ");
  } else if (ccValue >= 11) {
    ret.append("This program have a Low-Medium
Complexity ");
  } else {
    ret.append("This program have a Low Complexity ");
  }
  ret.append("\n");
  return ret;
}
```

Kode 4. 15 Fungsi-fungsi yang terdapat pada kelas *CyclomaticComplexity*

4.3.4 Kelas DepthOfInheritance

Kelas ini merupakan kelas yang berfungsi untuk melakukan perhitungan metrik *detpth of inheritance*. Pada kelas ini terdapat dua buah fungsi, yaitu fungsi *check* dan fungsi *getClassDIT*. Fungsi *check* berguna untuk melakukan proses perhitungan nilai metrik *depth of inheritance* dari inputan yang diberikan. Fungsi *getClassDIT* berguna untuk melakukan pengelompokan mengenai jenis dari metrik *depth of inheritance* dari berkas inputan tersebut.

```
public List<ClassData> check(BufferedReader br) {
  List<ClassData> classesDepth = new ArrayList<>();
  String line = null;
  try {
    line = br.readLine();
    while (line != null) {
      String[] arrayString = line.split(" ");
      for(int i=0; i<arrayString.length; i++){
        if(arrayString[i].equals("class")){
```

```
          ClassData cd = new ClassData();
          cd.setClassName(arrayString[i+1]);
          if (arrayString.length>(i+2)) {
            String x = arrayString[i+2];
            if(arrayString[i+2]!=null &&
arrayString[i+2].equals("extends")){
              cd.setValue(1);
              for(Object objCDE : classesDepth){
                ClassData cde = (ClassData)objCDE;
if(arrayString[i+3].contains(cde.getClassName())){

cd.setValue(cd.getValue()+cde.getValue());
                }
              }
            }
            classesDepth.add(cd);
          }
        }
      }
      line = br.readLine();
    }
  } catch (IOException e) {
    e.printStackTrace();
  }
  return (classesDepth);
}
public StringBuilder getClassDIT(BufferedReader br) {
  StringBuilder ret = new StringBuilder();
  List<ClassData> classDepths = check(br);
  ret.append("The Depth of Inheritance tree : ");
  ret.append("\n");
  ret.append("Result : ");
  ret.append("\n");
  for(Object objCDT : classDepths){
    ClassData cdt = (ClassData)objCDT;
    ret.append("For Class ")
```

```
      .append(cdt.getClassName())
      .append(" = ")
      .append(cdt.getValue());
ret.append("   , ");
    if (cdt.getValue()>=7) {
      ret.append("This Class have a High Complexity
");
```

```
    }else if (cdt.getValue()>5) {
      ret.append("This Class have a Medium-High
Complexity ");
    }else if (cdt.getValue()>3) {
      ret.append("This Class have a Low-Medium
Complexity ");
    }else {
      ret.append("This Class have a Low Complexity ");
    }
  ret.append("\n");
  inheritanceList.add(cdt);
  }
  return ret;
}
```

Kode 4. 16 Fungsi-fungsi yang terdapat pada kelas *DepthOfInheritance*

4.3.5 Kelas LineOfCode

Kelas ini merupakan kelas yang berfungsi untuk melakukan perhitungan metrik *line of code*. Pada kelas ini terdapat dua buah fungsi, yaitu fungsi *check* dan fungsi *getLineOfCode*. Fungsi *check* berguna untuk melakukan proses perhitungan nilai metrik *line of code* dari inputan yang diberikan. Fungsi *getLineOfCode* berguna untuk melakukan pengelompokan mengenai jenis dari metrik *line of code* dari berkas inputan tersebut.

```
public int check(BufferedReader br) {
  String line;
  int count=0;
  try {
    while((line=br.readLine())!=null){
      if (line.contains(";")) {
        count++;
      }
    }
    br.close();
  }
  catch (FileNotFoundException e) {
    System.err.println(e);
  }
```

```
   catch (IOException e) {
     System.err.println(e);
   }
   return count;
 }
 public StringBuilder getLineOfCode(BufferedReader br){
   StringBuilder ret = new StringBuilder();
   int locValue = check(br);
   ret.append("The Total Line of Code : ")
      .append(locValue);
   ret.append("\n");
   if (locValue>=150) {
     ret.append("This program have a Big size");
   }else if (locValue>=120) {
     ret.append("This program have a Medium-Big size");
   }else if (locValue>=90) {
     ret.append("This program have a Small-Medium
 size");
   }else if (locValue>=60) {
     ret.append("This program have a Small size");
   }
     ret.append("\n");
   return ret;
 }
```

Kode 4. 17 Fungsi-fungsi yang terdapat pada kelas *LineOfCode*

4.3.6 Kelas MyFileSize

Kelas ini merupakan kelas yang berfungsi untuk melakukan perhitungan metrik *file size*. Pada kelas ini terdapat satu buah fungsi, yaitu fungsi *check*. Fungsi *check* ini berguna untuk melakukan proses perhitungan nilai metrik *file size* dari inputan yang diberikan.

```
public StringBuilder check(String fileName){
  StringBuilder ret = new StringBuilder();
  File file = new File(fileName);
  if (file.exists()) {
    double bytes = file.length();
    double kiloBytes = (bytes/1024);
    NumberFormat nf = NumberFormat.getInstance();
    nf.setMinimumFractionDigits(2);
    nf.setMaximumFractionDigits(2);
```

```
    ret.append("The file size is ")
      .append(nf.format(kiloBytes));
    ret.append("\n");
  }
  return ret;
}
```

Kode 4. 18 Fungsi yang terdapat pada kelas *MyFileSize*

4.3.7 Kelas NumberOfChildren

Kelas ini merupakan kelas yang berfungsi untuk melakukan perhitungan metrik *number of children*. Pada kelas ini terdapat dua buah fungsi, yaitu fungsi *check* dan fungsi *getClassNOC*. Fungsi *check* berguna untuk melakukan proses perhitungan nilai metrik *number of children* dari inputan yang diberikan. Fungsi *getClassNOC* berguna untuk melakukan pengelompokan mengenai jenis dari metrik *number of children* dari berkas inputan tersebut.

```
public List<ClassData> check(BufferedReader br) {
  List<ClassData> parentClasses = new ArrayList<>();
  String line = null;
  try {
    line = br.readLine();
    while (line != null) {
      String[] arrayString = line.split(" ");
for(int i=0; i<arrayString.length; i++){
       if(arrayString[i].equals("class")){
         ClassData cp = new ClassData();
         cp.setClassName(arrayString[i+1]);
         cp.setValue(0);
         if (arrayString.length>(i+2)) {
           if(arrayString[i+2]!=null &&
arrayString[i+2].equals("extends")){
             for (int j = 0; j <
parentClasses.size(); j++) {
               ClassData cde = parentClasses.get(j);
               if
(arrayString[i+3].contains(cde.getClassName())) {
                 cde.setValue(cde.getValue()+1);
                 parentClasses.set(j, cde);
               }
```

```
                }
              }
            }
            parentClasses.add(cp);
          }
        }
        line = br.readLine();
      }
    } catch (IOException e) {
}
    return (parentClasses);
}
public StringBuilder getClassNOC(BufferedReader br) {
    List<ClassData> classDepths = check(br);
    StringBuilder ret = new StringBuilder();
    ret.append("The Number Of Children : ");
    ret.append("\n");
    for(Object objNOC : classDepths){
      ClassData noc = (ClassData)objNOC;
      ret.append("For Class ")
        .append(noc.getClassName())
        .append(" = ")
        .append(noc.getValue());
      ret.append("\n");
```

```
      if (noc.getValue()>=7) {
        ret.append("This Class have a High Complexity
");
      }else if (noc.getValue()>5) {
        ret.append("This Class have a Medium-High
Complexity ");
      }else if (noc.getValue()>3) {
        ret.append("This Class have a Low-Medium
Complexity ");
      }else {
        ret.append("This Class have a Low Complexity ");
      }
      ret.append("\n");
      childrenList.add(noc);
    }
    return ret;
}
```

Kode 4. 19 Fungsi-fungsi yang terdapat pada kelas *NumberOfChildren*

4.3.8 Kelas NumberOfSubunits

Kelas ini merupakan kelas yang berfungsi untuk melakukan perhitungan metrik *number of subunits*. Pada kelas ini terdapat dua buah fungsi, yaitu fungsi *check* dan fungsi *getSubUnits*. Fungsi *check* berguna untuk melakukan proses perhitungan nilai metrik *number of subunits* dari inputan yang diberikan. Fungsi *getSubUnits* berguna untuk melakukan pengelompokan mengenai jenis dari metrik *number of subunits* dari berkas inputan tersebut.

```
public List<ClassData> check(BufferedReader br) {
  List<ClassData> classes = new ArrayList<>();
  int classNum = -1;
  String cName="";
  String line = null;
  try {
    int bracketStart = 0;
    int bracketEnd = 0;

   boolean containProc = false;
    boolean startNew = false;
    line = br.readLine();
    while (line != null) {
      String[] arrayString = line.split(" ");
      for(int i=0; i<arrayString.length; i++){
        if (arrayString[i].contains("{")) {
          bracketStart = bracketStart+1;
          startNew = true;
        }
        if (arrayString[i].contains("}")) {
          bracketEnd = bracketEnd+1;
        }
        ClassData cd = new ClassData();
        if(arrayString[i].equals("class")){
          classNum = classNum+1;
          cName = arrayString[i+1];
          cd.setClassName(cName);
          cd.setValue(0);
```

```
            classes.add(cd);
          }else if (arrayString[i].equals("void") &&
!arrayString[i+1].contains("main")) {
            cd = classes.get(classNum);
            cd.setClassName(cName);
            cd.setValue(cd.getValue()+1);
            classes.set(classNum, cd);
          }
           if (bracketStart==bracketEnd &&
bracketStart>1) {
            startNew = false;
            bracketStart = 0;
            bracketEnd = 1;
            if (containProc) {
              cd = classes.get(classNum);
              cd.setClassName(cName);
              cd.setValue(cd.getValue()+1);
              classes.set(classNum, cd);
            }
          }
          if (startNew) {
            if (arrayString[i].equals("return")) {
              containProc = true;}
          }
        }
        line = br.readLine();
      }
    } catch (IOException e) {
      e.printStackTrace();
}
return (classes);
}
public StringBuilder getSubUnits(BufferedReader br) {
  StringBuilder ret = new StringBuilder();
  int subUnitsTotal = 0;
  List<ClassData> classes = check(br);
  ret.append("The Number of Sub unit is : ");
  for (ClassData classData : classes) {
    ret.append("Class ")
      .append(classData.getClassName())
      .append(" have ")
      .append(classData.getValue())
      .append(" function and procedures");
    ret.append(", ");
```

```
    if (classData.getValue()>=20) {
      ret.append("This Class have a High Complexity
");
    }else if (classData.getValue()>15) {
      ret.append("This Class have a Medium-High
Complexity ");
    }else if (classData.getValue()>12) {
      ret.append("This Class have a Low-Medium
Complexity ");
    }else {
      ret.append("This Class have a Low Complexity ");
    }
    subUnitsTotal = subUnitsTotal +
classData.getValue();
    subunitList.add(classData);
  }
  ret.append("\n");
  ret.append("Total SubUnits is ")
    .append(subUnitsTotal);
  ret.append("Number of subunits should be kept as low
as is reasonable");
  return ret;
}
```

Kode 4. 20 Fungsi-fungsi yang terdapat pada kelas *NumberOfSubunits*

4.3.9 Kelas ResponseClass

Kelas ini merupakan kelas yang berfungsi untuk melakukan perhitungan metrik *response for class*. Pada kelas ini terdapat tiga buah fungsi, yaitu fungsi *check, addListDistinctMethod,* dan fungsi *getResponseForClass*. Fungsi *check* berguna untuk melakukan proses perhitungan nilai dari jumlah *method* yang terdapat pada setiap kelas di dalam inputan yang diberikan. Fungsi *addListDistinctMethod* berguna untuk melakukan proses perhitungan jumlah dari pemanggilan method dari kelas lain yang digunakan pada masing-masing kelas yang diperiksa. Fungsi *getResponseForClass* berguna untuk menghitung nilai *response for class* dari masing-masing kelas

pada inputan dan melakukan pengelompokan mengenai jenis dari metrik *response for class* dari berkas inputan tersebut.

```
public int check(BufferedReader br) {
  int methodCount = 1;
  String[] keywords = {"void"};
  String words = "";
  String line = null;
  try {
    addListDistinctMethod(br);
    line = br.readLine();
    while (line != null) {
    StringTokenizer stTokenizer = new
StringTokenizer(line);
      while (stTokenizer.hasMoreTokens()) {
        words = stTokenizer.nextToken();
        boolean flagComment = false;
        for (int i = 0; i < keywords.length; i++) {
          if (keywords[i].equals("/*")){
            flagComment = true;
          }
          if(flagComment){
            if(keywords[i].equals("*/")){
              flagComment = false;
            }else{
              break;
            }
          }else if (keywords[i].equals("//")) {
            break;
          } else {
            if (keywords[i].equals(words) &&
!keywords[i+1].equals("main")) {
              methodCount++;
            }
          }
        }
      }
    }
  } catch (IOException e) {
    e.printStackTrace();
  }
  return (methodCount);
}
```

```
public void addListDistinctMethod(BufferedReader br){
String line;
try {
line = br.readLine(); // membaca baris pada file
while (line != null) { //lakukan looping selama data
baris pada file tidak kosong
String[] arrayString = line.split(" "); //pecah data
string pada line berdasarkan spasi (" ")

for(int i=0; i<arrayString.length; i++){ //lakukan
loop untuk setiap string pada baris
if (arrayString[i].equals("new")){ // memeriksa nilai
dari setiap array apakah pemanggilan class lain
(distinct class)
//menambahkan data pada distinct class
if (arrayString[i-1].equals("=")){
        distinctClass.add(arrayString[i-2]);
}else{
distinctClass.add(arrayString[i-1].replace("="," "));
}
}else if(distinctClass.size()>0){ // melakukan
pemeriksaan apakah distinct class sudah ada sebelumnya
for(int j=0;j<distinctClass.size();j++){// lakukan
looping untuk memeriksa semua data pada distinct class
String x = arrayString[i];
if(arrayString[i].contains(distinctClass.get(j)+".")){
//lakukan pemeriksaan apakah terdapat pemanggilan
method
String[] kw = x.split("\\(");

if(!distinctMethodKeywords.contains(kw[0])){//lakukan
pemeriksaan apakah distinct method yang sedang
diperiksa sudah ada pada list atau belum
distinctMethodKeywords.add(kw[0]);
}
}
}
}
}
line = br.readLine(); // membaca baris baru
}
} catch (IOException ex) {
```

```
Logger.getLogger(ResponseClass.class.getName()).log(Le
vel.SEVERE, null, ex);
}
}

public StringBuilder
getResponseForClass(BufferedReader br) { // method
untuk menampilkan data hasil perhitungan
StringBuilder ret = new StringBuilder();
int wmValue = check(br);
int total = wmValue+distinctMethodKeywords.size();
ret.append("The Response For Class number is :")
.append(total);
ret.append("\n");
ret.append("Result : The Method Count = ")
.append(wmValue)
.append(" and the Distinct Method = ")
.append(distinctMethodKeywords.size())
.append(", ");

if (total>=100) {
ret.append("This program have a High Complexity ");
}else if(total>=70){
ret.append("This program have a Medium-High Complexity
");
}else if(total>=50){
ret.append("This program have a Low-Medium Complexity
");
}else{
ret.append("This program have a Low Complexity ");
}

ret.append("\n");
return ret;
}
```

Kode 4. 21 Fungsi-fungsi yang terdapat pada kelas *ResponseClass*

4.3.10 Kelas WeightedMethod

Kelas ini merupakan kelas yang berfungsi untuk melakukan perhitungan metrik *weighted method*. Pada kelas ini terdapat dua buah fungsi, yaitu fungsi *check* dan fungsi *getWeightedMethod*. Fungsi *check* berguna untuk melakukan

proses perhitungan nilai metrik *weighted method* dari inputan yang diberikan. Fungsi *getWeightedMethod* berguna untuk melakukan pengelompokan mengenai jenis dari metrik *weighted method* dari berkas inputan tersebut.

```
public List<ClassData> check(BufferedReader br) {
  List<ClassData> classes = new ArrayList<>();
  int classNum = -1;
  String cName="";
  String line = null;
  try {
    line = br.readLine();
    while (line != null) {
      String[] arrayString = line.split(" ");
      for(int i=0; i<arrayString.length; i++){
        ClassData cd = new ClassData();
        if(arrayString[i].equals("class")){
          classNum = classNum+1;
          cName = arrayString[i+1];
          cd.setClassName(cName);
          cd.setValue(0);
          classes.add(cd);
        }else if (arrayString[i].equals("void") &&
!arrayString[i+1].contains("main")) {
          cd = classes.get(classNum);
          cd.setClassName(cName);
          cd.setValue(cd.getValue()+1);
          classes.set(classNum, cd);
        }
      }
      line = br.readLine();
    }
  } catch (IOException e) {
    e.printStackTrace();
  }
  return (classes);
}

public StringBuilder getWeightedMethod(BufferedReader
br) {
  StringBuilder ret = new StringBuilder();
  int wmcTotal = 0;
  List<ClassData> classes = check(br);
```

```
  ret.append("The Weighted Method is : ");
  for (ClassData classData : classes) {
    ret.append("\n");
    ret.append("Class ")
      .append(classData.getClassName())
      .append(" have ")
      .append(classData.getValue())
      .append(" method count");
      wmcTotal = wmcTotal + classData.getValue();
      methodList.add(classData);
    }
    ret.append("\n");
    if (wmcTotal>=50) {
      ret.append("This program have a High Complexity,
For the total WMC = ")
        .append(wmcTotal);
    }else if (wmcTotal>=35) {
      ret.append("This program have a Medium-High
Complexity, For the total WMC = ")
        .append(wmcTotal);
    }else if (wmcTotal>=20) {
      ret.append("This program have a Low-Medium
Complexity, For the total WMC = ")
        .append(wmcTotal);
    }else {
      ret.append("This program have a Low Complexity,
For the total WMC = ")
        .append(wmcTotal);
    }
    ret.append("\n");
    ret.append("WMC should be kept as low as is
reasonable");
    return ret;
}
```

Kode 4. 22 Fungsi-fungsi yang terdapat pada kelas *WeightedMethod*

4.4 Implementasi Lapisan Model

Lapisan ini berisi semua atribut yang diperlukan untuk menyimpan elemen-elemen yang diperlukan untuk baik dalam melakukan kalkulasi metrik maupun untuk data yang akan ditampilkan pada pdf. Lapisan data menyimpan atribut yang

diperlukan oleh kelas-kelas pada lapisan *control* dan berisi atribut yang bersifat *private*.

4.4.1 Kelas *ClassData*

Kelas ini berisi atribut yang diperlukan untuk merepresentasikan data dari setiap kelas yang terdapat pada inputan nantinya. Kelas ini berguna sebagai tipe data yang digunakan untuk *parsing* data antar kelas. Pada kelas ini terdapat dua buah atribut, yaitu *className* yang bertipe *string* dan *value* yang bertipe *int*. Pada kelas ini juga terdapat *setter* dan *getter* standar guna mengakses atribut yang terdapat di dalamnya.

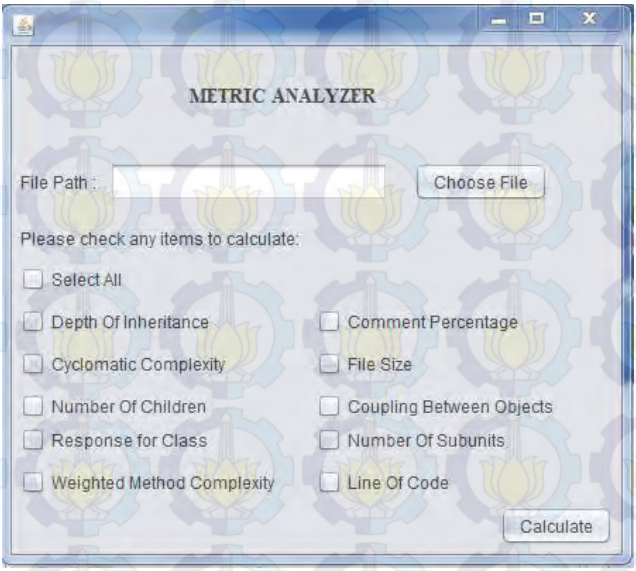
4.4.2 Kelas *PdfData*

Kelas ini berisi atribut yang diperlukan untuk menampung data hasil kalkulasi yang nantinya akan ditampilkan pada berkas hasil akhir. Kelas ini berguna sebagai tipe data yang nantinya akan digunakan sebagai perantara baik untuk menghasilkan *chart* hasil kalkulasi maupun untuk menampilkan hasil kalkulasi. Pada kelas ini terdapat dua buah atribut, yakni *data* yang bertipe *StringBuilder* dan *type* yang bertipe *String*. Pada kelas ini juga terdapat *setter* dan *getter* standar guna mengakses atribut yang terdapat di dalamnya.

4.5 Implementasi Antarmuka Pengguna

Implementasi tampilan antarmuka pengguna dibangun dengan menggunakan standar *swing* dari java. Berikut ini akan dijelaskan mengenai tampilan antarmuka yang terdapat pada kakas bantu ini.

4.5.1 Tampilan Utama



Gambar 4. 1 Tampilan utama kakas bantu

Tampilan utama dari kakas bantu ini dapat dilihat pada Gambar 4. 1. Pada halaman ini pengguna dapat menekan tombol *Choose File* untuk memilih data berkas masukan yang ingin dikalkulasi. Setelah pengguna memilih berkas inputan, alamat dari berkas tersebut akan muncul pada kotak teks yang tersedia. Pengguna lalu dapat memilih jenis metrik yang ingin dihitung dengan mencentang kotak-kotak yang disediakan sesuai dengan nama metrik yang ingin dihitung. Setelah itu pengguna dapat menekan tombol *calculate* agar kakas bantu ini dapat menghitung nilai metrik dari berkas inputan yang dipilih, dan data hasil akhirnya akan disimpan pada sebuah berkas pdf dan akan ditampilkan.

BAB V
PENGUJIAN DAN EVALUASI

Bab ini membahas pengujian dan evaluasi pada kakas bantu yang dikembangkan. Pengujian yang dilakukan adalah pengujian terhadap kebutuhan fungsionalitas sistem dan kegunaan sistem. Pengujian fungsionalitas mengacu pada kasus penggunaan pada bab tiga. Pengujian kegunaan program dilakukan dengan mengetahui tanggapan dari pengguna terhadap sistem. Hasil evaluasi menjabarkan tentang rangkuman hasil pengujian pada bagian akhir bab ini.

5.1 Lingkungan Pengujian

Lingkungan pengujian sistem pada pengerjaan tugas akhir ini dilakukan pada lingkungan dan alat kakas sebagai berikut:

Prosesor	: Intel Core i7-2630QM CPU @ 2.00GHz
Memori	: 4.00 GB
Jenis Device	: Laptop
Sistem Operasi	: Microsoft Windows 7 Enterprise 32 bit
IDE NetBeans	: NetBeans IDE 8.0.2

5.2 Skenario Pengujian

Pada bagian ini akan dijelaskan tentang skenario pengujian yang dilakukan. Pengujian yang dilakukan adalah pengujian kebutuhan fungsionalitas. Pengujian fungsionalitas menggunakan metode kotak hitam (*black box*). Metode ini menekankan pada kesesuaian hasil keluaran sistem.

Pengujian terhadap hasil deteksi dilakukan dengan membsandingkan hasil deteksi sistem dengan hasil analisa manual yang dilakukan oleh pengembang aplikasi. Pengembang aplikasi melakukan penghitungan manual setiap metrik untuk setiap kasus percobaan. Hasil perhitungan metrik secara manual tersebut kemudian dibandingkan dengan hasil perhitungan dari

system untuk melihat kesesuaian hasil perhitungan kakas bantu yang dibangun.

5.2.1 Pengujian Fungsionalitas

Pengujian fungsionalitas sistem dilakukan dengan menyiapkan sejumlah skenario sebagai tolak ukur keberhasilan pengujian. Pengujian pada kebutuhan fungsionalitas dapat dijabarkan pada subbab berikut.

5.2.1.1 Pengujian Fitur Menampilkan Hasil Kalkulasi Metrik

Pengujian fitur menampilkan rekomendasi ekstraksi fungsi pada kode sumber dilakukan dengan menggunakan beberapa data uji. Data uji tersebut berupa tiga *source code* yang dikembangkan menggunakan Bahasa pemrogramn Java. Data aplikasi yang digunakan dalam pengujian dapat dilihat pada Tabel 5. 1. Nilai dari kalkulasi metrik yang terdapat pada tabel merupakan hasil kalkulasi yang dianalisa pengembang sebagai hasil dari perhitungan yang dilakukan oleh kakas bantu yang dibangun. Pada Gambar 5. 1 sampai Gambar 5. 7 adalah hasil printout pengujian perangkat lunak dengan menggunakan skenario 1

Tabel 5. 1 Daftar Aplikasi untuk Data Uji

Nama Aplikasi	Ukuran Aplikasi	
	Jumlah Kelas	Jumlah *Method*
Employments	5	15
TronGames	5	25
TicTacToe	6	42
OSM	17	67

Tabel 5. 2 Pengujian Fitur Menampilkan Hasil Kalkulasi Metrik Skenario 1

ID	UJ.UC-0001
Referensi Kasus Penggunaan	UC-0001
Nama	Pengujian fitur menampilkan hasil kalkulasi metrik pada *source code* aplikasi
Tujuan Pengujian	Menguji fitur untuk melakukan penghitungan seluruh nilai metrik pada applikasi.
Skenario 1	**Data kode sumber menggunakan data kode sumber aplikasi pertama.**
Kondisi Awal	Kakas bantu dalam keadaan aktif. *Source code* applikasi yang akan diuji telah dipilih pada kakas bantu sebagai sumber *input.*
Data Uji	**Data uji diperoleh dari kode sumber aplikasi pertama.**
Langkah Pengujian	Pengguna memilih untuk melakukan penghitungan seluruh nilai metrik dengan mencentang seluruh metrik. Kemudian pengguna menekan tombol *Calculate.*
Hasil Yang Diharapkan	Daftar hasil kalkulasi untuk semua metrik dari *source code* ditampilkan pada file dengan format .pdf.
Hasil Yang Didapat	Daftar hasil kalkulasi untuk semua metrik dari *source code* ditampilkan pada file dengan format .pdf.
Hasil Pengujian	Berhasil
Kondisi Akhir	Daftar hasil kalkulasi dapat dilihat pada Tabel 5. 3. Tampilan hasil kalkulasi dapat dilihat pada file .pdf terlampir.

Tabel 5. 3 Hasil Uji Skenario 1

Metrik	Kelas	Nilai Tools	Nilai Manual
Depth of Inheritance	Employee	0	0
	GeneralManager	1	1
	Engineer	1	1
	Worker	1	1
	HourlyEmployee	2	2
Cyclomatic Complexity		7	7
Number of Children	Employee	3	3
	General Manager	0	0
	Engineer	0	0
	Worker	1	1
	Hourly Employee	0	0
Response For Class	Employee	3	3
	General Manager	5	5
	Engineer	4	4
	Worker	5	5
	Hourly Employee	7	7
Weighted Method for Class	Employee	3	3
	General Manager	3	3
	Engineer	2	2
	Worker	3	3
	Hourly Employee	3	3
Comment Percentage		0	0
Filesize		3.99 kb	4 kb
Coupling Between Object	Employee	3	3
	General Manager	3	3
	Engineer	1	1
	Worker	3	3
	Hourly Employee	2	2
Number of Subunits	Employee	3	3
	General Manager	3	3
	Engineer	2	2
	Worker	3	3
	Hourly Employee	3	3
Line of Code		114	114

Metric Analyzer

Depth Of Inheritance

The Depth of Inheritance tree :

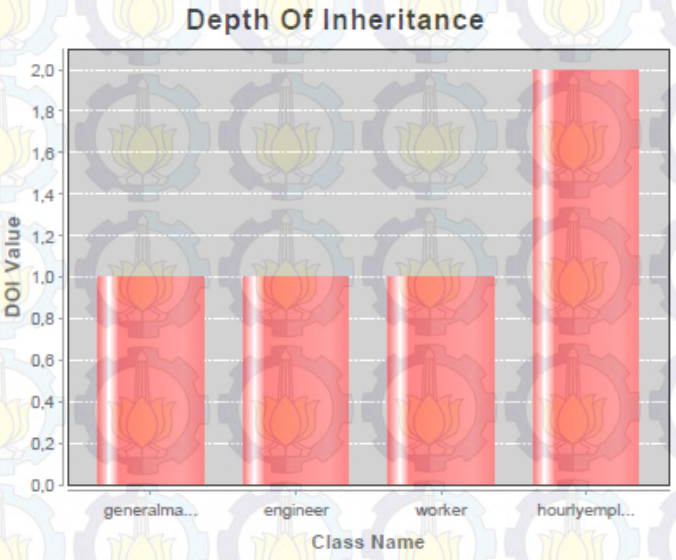
Result :

For Class generalmanager = 1 , This Class have a Low Complexity

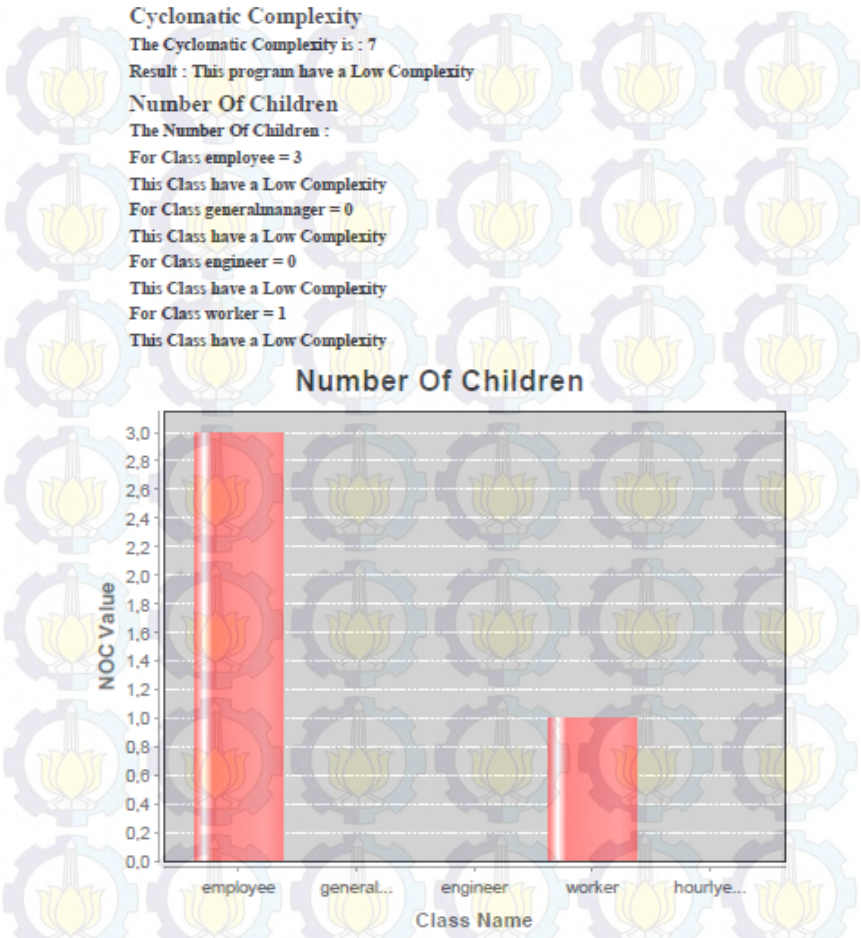
For Class engineer = 1 , This Class have a Low Complexity

For Class worker = 1 , This Class have a Low Complexity

For Class hourlyemployee = 2 , This Class have a Low Complexity



Gambar 5. 1 Hasil Perhitungam DIT untuk Skenario 1 pada Aplikasi



Gambar 5. 2 Hasil Perhitungan NOC untuk Skenario 1 pada Aplikasi

Response for Class

The Response For Class number is :

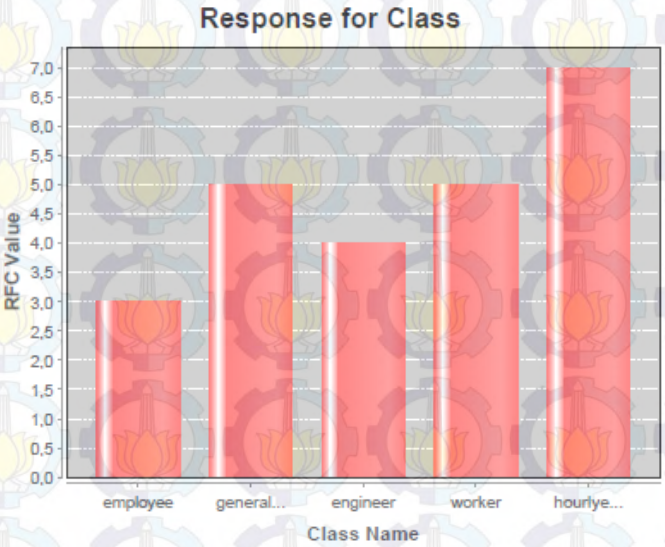
Class employee have 3 operations, This Class have a Low Complexity

Class generalmanager have 5 operations, This Class have a Low Complexity

Class engineer have 4 operations, This Class have a Low Complexity

Class worker have 5 operations, This Class have a Low Complexity

Class hourlyemployee have 7 operations, This Class have a Low Complexity Respoonse for class value should be kept as low as is reasonable



Gambar 5. 3 Hasil Perhitungam RFC untuk Skenario 1 pada Aplikasi

Weighted Method for Class

The Weighted Method is :

Class employee have 3 method count

Class generalmanager have 3 method count

Class engineer have 2 method count

Class worker have 3 method count

Class hourlyemployee have 3 method count

This program have a Low Complexity, For the total WMC = 14

WMC should be kept as low as is reasonable

Gambar 5. 4 Hasil Perhitungam WMC untuk Skenario 1 pada Aplikasi

Comment Percentage

The Comment Percentage is : 10

This program have a Small size

File Size

The file size is 3,12

Coupling Between Objects

The Coupling Between Objects value :

For Class employee = 3

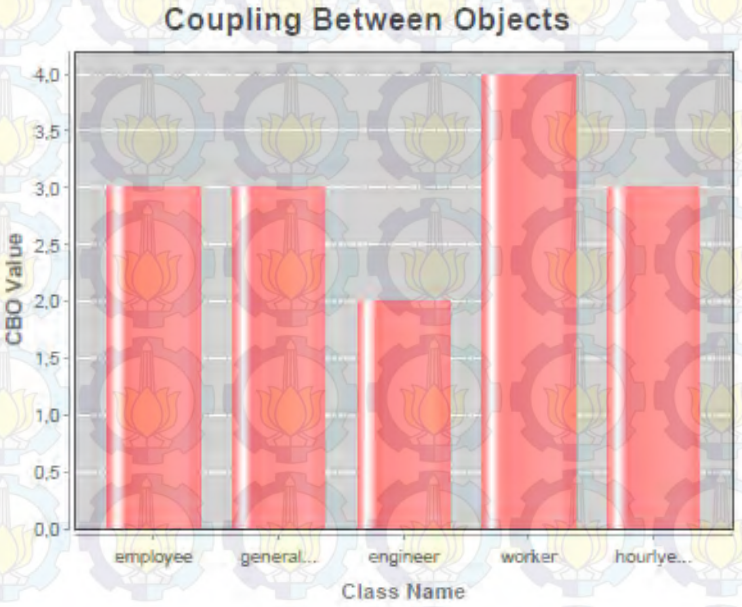
For Class generalmanager = 3

For Class engineer = 2

For Class worker = 4

For Class hourlyemployee = 3

This program have a Low-Medium Coupling, For the total CBO = 15



Gambar 5. 5 Hasil Perhitungam Comment Percentage, File Size dan CBO untuk Skenario 1 pada Aplikasi

Number of Subunits

The Number of Sub unit is :

Class employee have 3 function and procedures, This Class have a Low Complexity

Class generalmanager have 3 function and procedures, This Class have a Low Complexity

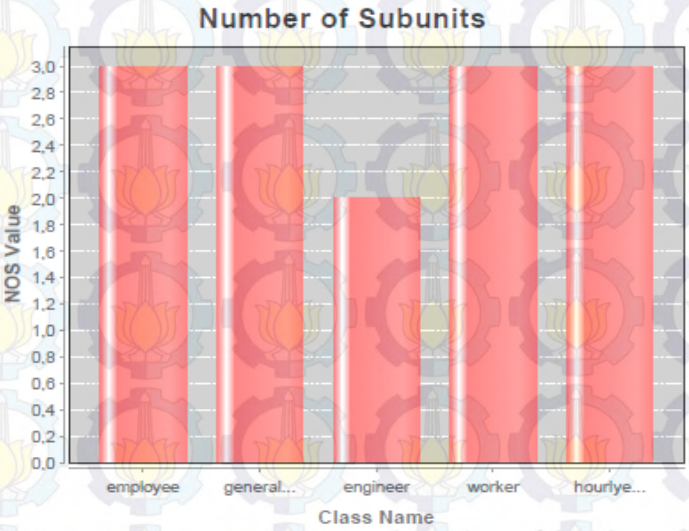
Class engineer have 2 function and procedures, This Class have a Low Complexity

Class worker have 3 function and procedures, This Class have a Low Complexity

Class hourlyemployee have 3 function and procedures, This Class have a Low Complexity

Total SubUnits is 14

Number of subunits should be kept as low as is reasonable



Line Of Code

The Total Line of Code : 111

This program have a Small-Medium size

Gambar 5. 6 Hasil Perhitungam Number of Subuntis dan Line of Code untuk Skenario 1 pada Aplikasi

Final Conclussion

Over all of the metric calculation, we assume that the tested source code is having a LOW complexity

LEGEND

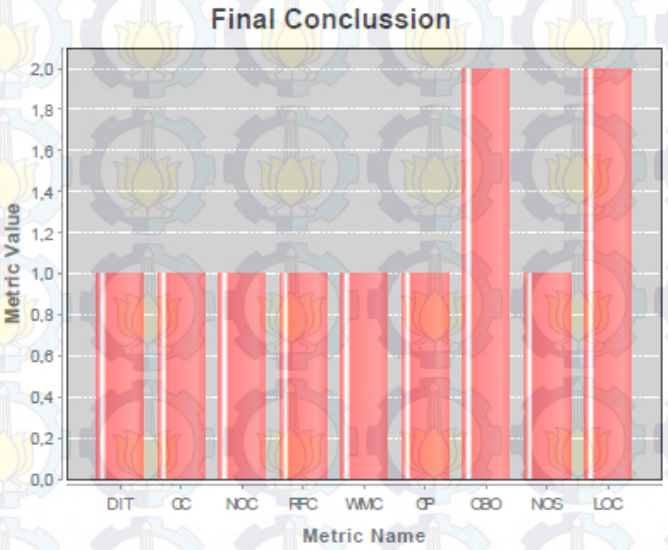
Explanation of 'y' axis:

Value 1 = Low Risk

Value 2 = Low-Medium Risk

Value 3 = Medium-High Risk

Value 4 = High Risk



Gambar 5. 7 Hasil Perhitungam Final untuk Skenario 1 pada Aplikasi

Untuk file hasil data pengujian 1 dapat dilihat pada lampiran. Skenario pengujian fungsionalitas sistem dapat dilihat pada Tabel 5. 2. Hasil pengujian untuk skenario 1 dapat dilihat

pada Tabel 5. 3. Tabel tersebut berisi hasil dari kalkulasi untuk setiap metrik dengan berkas masukan dari *source code* pertama. Untuk data uji dan hasil pengujian dapat dilihat pada berkas lampiran.

Tabel 5. 4 Pengujian Fitur Menampilkan Hasil Kalkulasi Metrik Skenario 2

ID	UJ.UC-0002
Referensi Kasus Penggunaan	UC-0002
Nama	Pengujian fitur menampilkan hasil kalkulasi metrik pada *source code* aplikasi
Tujuan Pengujian	Menguji fitur untuk melakukan penghitungan seluruh nilai metrik pada applikasi.
Skenario 2	**Data kode sumber menggunakan data kode sumber aplikasi kedua.**
Kondisi Awal	Kakas bantu dalam keadaan aktif. *Source code* applikasi yang akan diuji telah dipilih pada kakas bantu sebagai sumber *input.*
Data Uji	**Data uji diperoleh dari kode sumber aplikasi kedua.**
Langkah Pengujian	Pengguna memilih untuk melakukan penghitungan seluruh nilai metrik dengan mencentang seluruh metrik. Kemudian pengguna menekan tombol *Calculate.*
Hasil Yang Diharapkan	Daftar hasil kalkulasi untuk semua metrik dari *source code* ditampilkan pada file dengan format .pdf.
Hasil Yang Didapat	Daftar hasil kalkulasi untuk semua metrik dari *source code* ditampilkan pada file dengan format .pdf.
Hasil Pengujian	Berhasil

ID	UJ.UC-0002
Kondisi Akhir	Daftar hasil kalkulasi dapat dilihat pada Tabel 5. 5. Tampilan hasil kalkulasi dapat dilihat pada file .pdf terlampir.

Tabel 5. 5 Hasil Uji Skenario 2

Metrik	Kelas	Nilai Tools	Nilai Manual
Depth of Inheritance	Tron	1	1
	Core	0	0
	ScreenManager	0	0
	Animation	0	0
	oneScene	0	0
Cyclomatic Complexity		84	84
Number of Children	Tron	0	0
	Core	0	0
	ScreenManager	0	0
	Animation	0	0
	oneScene	0	0
Response For Class	Tron	14	14
	Core	6	6
	ScreenManager	3	3
	Animation	3	3
	oneScene	0	0
Weighted Method for Class	Tron	12	12
	Core	6	6
	ScreenManager	3	3
	Animation	3	3
	oneScene	0	0
Comment Percentage		0	0
Filesize		10.05 kb	11 kb
Coupling Between Object	Tron	1	1
	Core	1	1
	ScreenManager	1	1
	Animation	0	0
	oneScene	0	0
Number of Subunits	Tron	12	12
	Core	6	6
	ScreenManager	3	3

Metrik	Kelas	Nilai Tools	Nilai Manual
	Animation	3	3
	oneScene	0	0
Line of Code		362	362

Untuk file hasil data pengujian 2 dapat dilihat pada lampiran. Skenario pengujian fungsionalitas sistem dapat dilihat pada Tabel 5. 4. Hasil pengujian untuk skenario 2 dapat dilihat pada Tabel 5. 5. Tabel tersebut berisi hasil dari kalkulasi untuk setiap metrik dengan berkas masukan dari *source code* kedua. Untuk data uji dan hasil pengujian dapat dilihat pada berkas lampiran.

Tabel 5. 6 Pengujian Fitur Menampilkan Hasil Kalkulasi Metrik Skenario 3

ID	UJ.UC-0003
Referensi Kasus Penggunaan	UC-0003
Nama	Pengujian fitur menampilkan hasil kalkulasi metrik pada *source code* aplikasi
Tujuan Pengujian	Menguji fitur untuk melakukan penghitungan seluruh nilai metrik pada applikasi.
Skenario 3	**Data kode sumber menggunakan data kode sumber aplikasi ketiga.**
Kondisi Awal	Kakas bantu dalam keadaan aktif. *Source code* applikasi yang akan diuji telah dipilih pada kakas bantu sebagai sumber *input.*
Data Uji	**Data uji diperoleh dari kode sumber aplikasi ketiga.**
Langkah Pengujian	Pengguna memilih untuk melakukan penghitungan seluruh nilai metrik dengan mencentang seluruh metrik. Kemudian pengguna menekan tombol *Calculate.*

ID	UJ.UC-0003
Hasil Yang Diharapkan	Daftar hasil kalkulasi untuk semua metrik dari *source code* ditampilkan pada file dengan format .pdf.
Hasil Yang Didapat	Daftar hasil kalkulasi untuk semua metrik dari *source code* ditampilkan pada file dengan format .pdf.
Hasil Pengujian	Berhasil
Kondisi Akhir	Daftar hasil kalkulasi dapat dilihat pada Tabel 5. 7. Tampilan hasil kalkulasi dapat dilihat pada file .pdf terlampir.

Tabel 5. 7 Hasil Uji Skenario 3

Metrik	Kelas	Nilai Tools	Nilai Manual
Depth of Inheritance	TicTacToe	1	1
	Player	0	0
	Human	1	1
	Computer	1	1
	Node	0	0
	Layer	0	0
Cyclomatic Complexity		72	72
Number of Children	TicTacToe	0	0
	Player	2	2
	Human	0	0
	Computer	0	0
	Node	0	0
	Layer	0	0
Response For Class	TicTacToe	25	25
	Player	5	5
	Human	2	2
	Computer	13	13
	Node	10	10
	Layer	1	1
Weighted Method for Class	TicTacToe	23	23
	Player	4	4
	Human	1	1
	Computer	7	7

Metrik	Kelas	Nilai Tools	Nilai Manual
	Node	5	5
	Layer	1	1
Comment Percentage		59	59
Filesize		10.05 kb	11 kb
Coupling Between Object	TicTacToe	1	1
	Player	4	4
	Human	1	1
	Computer	2	2
	Node	10	10
	Layer	2	2
Line of Code		362	362
Number of Subunits	TicTacToe	24	24
	Player	5	5
	Human	2	2
	Computer	13	13
	Node	10	10
	Layer	1	1

Untuk file hasil data pengujian 3 dapat dilihat pada lampiran. Skenario pengujian fungsionalitas sistem dapat dilihat padaTabel 5. 6. Hasil pengujian untuk skenario 3 dapat dilihat pada Tabel 5. 7. Tabel tersebut berisi hasil dari kalkulasi untuk setiap metrik dengan berkas masukan dari *source code* kedua. Untuk data uji dan hasil pengujian dapat dilihat pada berkas lampiran.

Tabel 5. 8 Pengujian Fitur Menampilkan Hasil Kalkulasi Metrik Skenario 4

ID	UJ.UC-0004
Referensi Kasus Penggunaan	UC-0004
Nama	Pengujian fitur menampilkan hasil kalkulasi metrik pada *source code* aplikasi
Tujuan Pengujian	Menguji fitur untuk melakukan penghitungan seluruh nilai metrik pada applikasi.

ID	UJ.UC-0004
Skenario 4	**Data kode sumber menggunakan data kode sumber aplikasi keempat.**
Kondisi Awal	Kakas bantu dalam keadaan aktif. *Source code* applikasi yang akan diuji telah dipilih pada kakas bantu sebagai sumber *input.*
Data Uji	**Data uji diperoleh dari kode sumber aplikasi keempat.**
Langkah Pengujian	Pengguna memilih untuk melakukan penghitungan seluruh nilai metrik dengan mencentang seluruh metrik. Kemudian pengguna menekan tombol *Calculate.*
Hasil Yang Diharapkan	Daftar hasil kalkulasi untuk semua metrik dari *source code* ditampilkan pada file dengan format .pdf.
Hasil Yang Didapat	Daftar hasil kalkulasi untuk semua metrik dari *source code* ditampilkan pada file dengan format .pdf.
Hasil Pengujian	Berhasil
Kondisi Akhir	Daftar hasil kalkulasi dapat dilihat pada Tabel 5. 9. Tampilan hasil kalkulasi dapat dilihat pada file .pdf terlampir.

Tabel 5. 9 Hasil Uji Skenario 4

Metrik	Kelas	Nilai Tools	Nilai Manual
Depth of Inheritance	TileController	0	0
	Tile	0	0
	Style	0	0
	OsmTileLoader	0	0
	implements	0	0
	OsmMercator	0	0
	OsmFileCacheTileLoader	1	1
	FileLoadJob	0	0
	MemoryTileCache	0	0

Metrik	Kelas	Nilai Tools	Nilai Manual
	CacheEntry	0	0
	CacheLinkedListElement	0	0
	MapRectangleImpl	1	1
	MapPolygonImpl	1	1
	OsmTileSource	0	0
	Mapnik	1	1
	CycleMap	1	1
	Demo	1	1
Cyclomatic Complexity		328	328
Number of Children	TileController	0	0
	Tile	3	3
	Style	0	0
	OsmTileLoader	1	1
	implements	0	0
	OsmMercator	0	0
	OsmFileCacheTileLoader	0	0
	FileLoadJob	0	0
	MemoryTileCache	0	0
	CacheEntry	0	0
	CacheLinkedListElement	0	0
	MapRectangleImpl	0	0
	MapPolygonImpl	0	0
	OsmTileSource	2	2
	Mapnik	0	0
	CycleMap	0	0
	Demo	0	0
Response For Class	TileController	5	5
	Tile	14	14
	Style	6	6
	OsmTileLoader	5	5
	implements	0	0
	OsmMercator	0	0
	OsmFileCacheTileLoader	0	0
	FileLoadJob	12	12
	MemoryTileCache	5	5
	CacheEntry	0	0
	CacheLinkedListElement	4	4
	MapRectangleImpl	1	1
	MapPolygonImpl	2	2

Metrik	Kelas	Nilai Tools	Nilai Manual
	OsmTileSource	0	0
	Mapnik	0	0
	CycleMap	0	0
	Demo	14	14
Weighted Method for Class	TileController	4	4
	Tile	9	9
	Style	6	6
	OsmTileLoader	3	3
	implements	0	0
	OsmMercator	0	0
	OsmFileCacheTileLoader	0	0
	FileLoadJob	11	11
	MemoryTileCache	5	5
	CacheEntry	0	0
	CacheLinkedListElement	4	4
	MapRectangleImpl	1	1
	MapPolygonImpl	2	2
	OsmTileSource	0	0
	Mapnik	0	0
	CycleMap	0	0
	Demo	13	13
Comment Percentage		211	211
Filesize		70.99 kb	70.9 kb
Coupling Between Object	TileController	0	0
	Tile	15	15
	Style	5	5
	OsmTileLoader	1	1
	implements	0	0
	OsmMercator	0	0
	OsmFileCacheTileLoader	1	1
	FileLoadJob	0	0
	MemoryTileCache	0	0
	CacheEntry	11	11
	CacheLinkedListElement	1	1
	MapRectangleImpl	0	0
	MapPolygonImpl	0	0
	OsmTileSource	0	0

Metrik	Kelas	Nilai Tools	Nilai Manual
	Mapnik	0	0
	CycleMap	0	0
	Demo	0	0
Line of Code		1378	1378
Number of Subunits	TileController	5	5
	Tile	14	14
	Style	6	6
	OsmTileLoader	5	5
	implements	0	0
	OsmMercator	0	0
	OsmFileCacheTileLoader	0	0
	FileLoadJob	12	12
	MemoryTileCache	5	5
	CacheEntry	0	0
	CacheLinkedListElement	4	4
	MapRectangleImpl	1	1
	MapPolygonImpl	2	2
	OsmTileSource	0	0
	Mapnik	0	0
	CycleMap	0	0
	Demo	13	13

Untuk file hasil data pengujian 3 dapat dilihat pada lampiran. Skenario pengujian fungsionalitas sistem dapat dilihat pada Tabel 5. 8. Hasil pengujian untuk skenario 3 dapat dilihat pada Tabel 5. 9. Tabel tersebut berisi hasil dari kalkulasi untuk setiap metrik dengan berkas masukan dari *source code* kedua. Untuk data uji dan hasil pengujian dapat dilihat pada berkas lampiran.

5.3 Evaluasi Pengujian

Pada subbab ini akan diberikan hasil evaluasi dari pengujian-pengujian yang telah dilakukan. Evaluasi yang diberikan meliputi evaluasi pengujian penghitungan nilai setiap metrik dari *source code* yang diberikan.

5.3.1 Evaluasi Pengujian Fungsionalitas

Rangkuman mengenai hasil pengujian fungsionalitas dapat dilihat pada Tabel 5. 10. Berdasarkan data pada tabel tersebut, semua skenario pengujian berhasil dan program berjalan dengan baik. Sehingga bisa ditarik disimpulkan bahwa fungsionalitas dari program telah bisa bekerja sesuai dengan yang diharapkan.

Tabel 5. 10 Rangkuman Hasil Pengujian

ID	Nama	Skenario	Hasil
UJ.UC-0001	Pengujian fitur menghitung nilai metrik.	Skenario 1	Berhasil
		Skenario 2	Berhasil
		Skenario 3	Berhasil

5.3.2 Pengujian dengan Metode Kuesioner

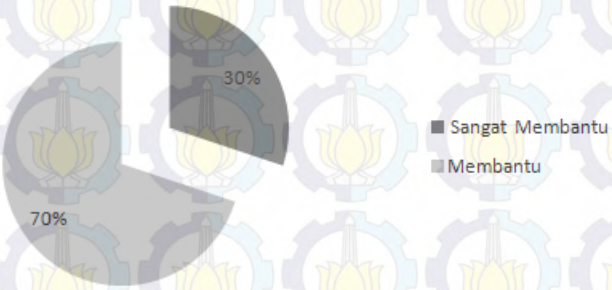
Dari hasil penelitian yang berupa kuesioner untuk orang yang berpengalaman di bidang pemrograman terutama dengan menggunakan bahasa pemrograman Java. Adapun profil penguji adalah sebagai berikut ini :

Tabel 5. 11 Profil Penguji pada Kuesioner

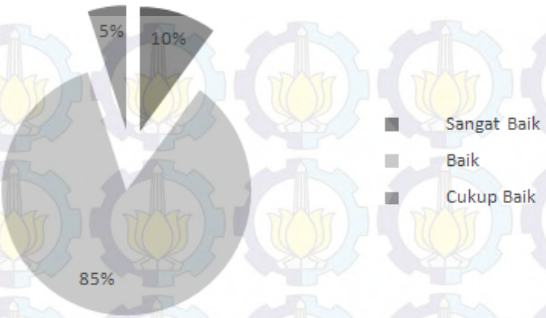
No	Kategori Responden	Pilihan	Jumlah
1	Jenis Kelamin	Laki-Laki	12
		Perempuan	8
2	**Usia**	**<20 tahun**	**-**
		20-25 tahun	5
		25-30 tahun	15
		>30 tahun	-
3	Pendidikan	SMA	-
		Diploma	5
		Sarjana	15

No	Kategori Responden	Pilihan	Jumlah
		Pascasarjana	-
4	Lama Pengalaman Pemrograman	<3 tahun	7
		3-5 tahun	3
		>5 tahun (expert)	10
5	Familiar dengan bahasa pemrograman Java	Ya	20
		Tidak	-

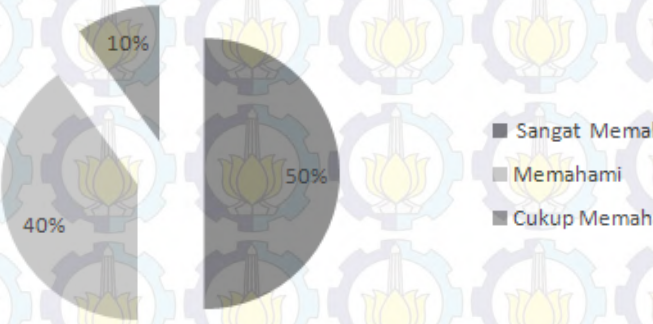
Hasil pengujian perangkat lunak ini dirasa membantu dan mudah penggunaanya. Tampilan dari aplikasi ini juga dirasa baik dan mudah dimengerti. Hasil dari rekapitulasi ini direpresentasikan dalam bentuk bagan-bagan berikut:



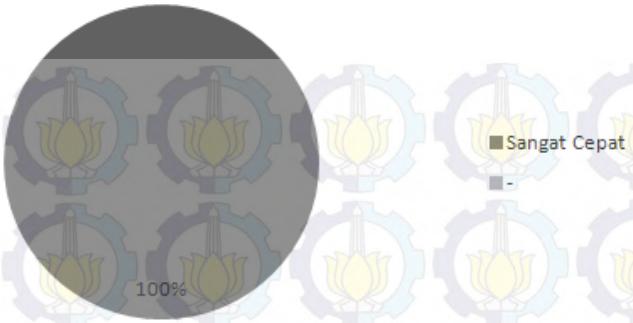
Gambar 5. 8 Hasil perhitungan sejauh mana aplikasi ini membantu proses pengukuran kualitas perangkat lunak.



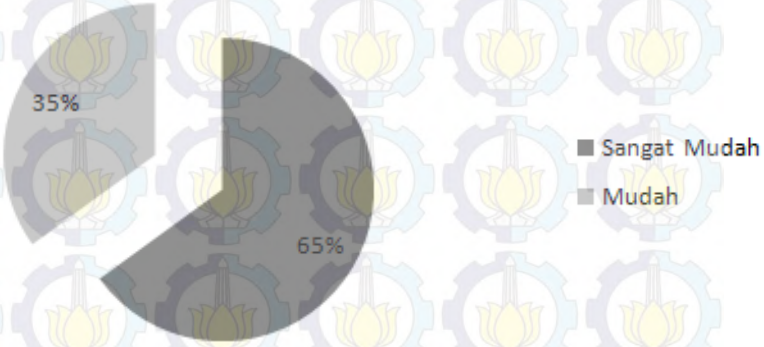
Gambar 5. 9 Hasil perhitungan pendapat mengenai tampilan awal pada aplikasi



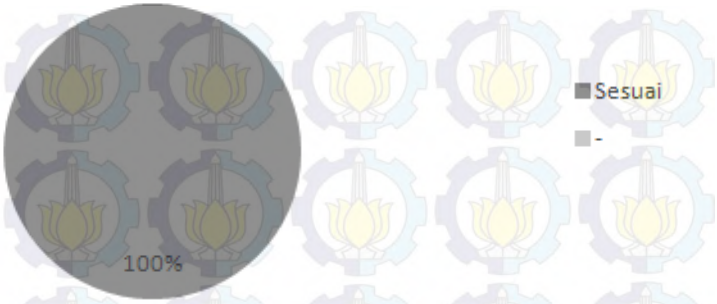
Gambar 5. 10 Hasil perhitungan sejauh mana tampilan akhir aplikasi dapat dipahami



Gambar 5. 11 Hasil perhitungan pendapat mengenai waktu yang dibutuhkan aplikasi ketika dijalankan

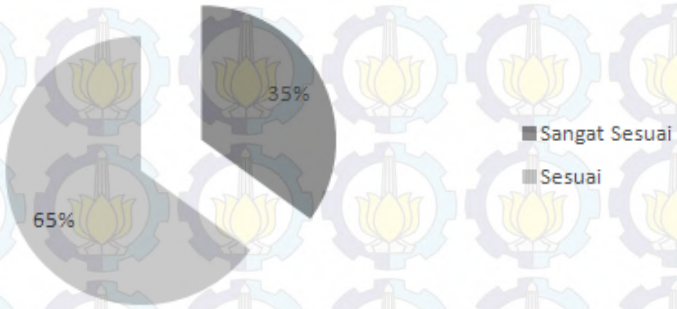


Gambar 5. 12 Hasil perhitungan kemudahan aplikasi untuk digunakan

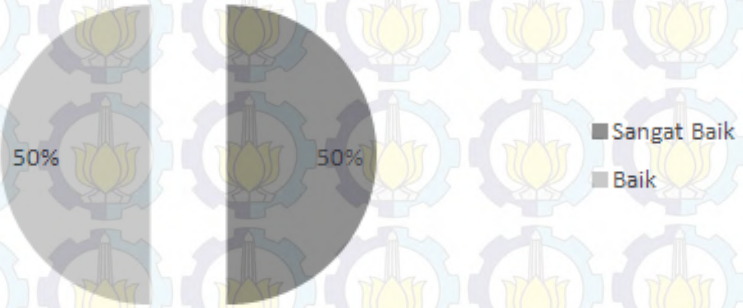


Gambar 5. 13 Hasil perhitungan mengenai kesesuaian warna, ukuran dan pemilhan jenis tulisan pada tampilan dan hasil pada aplikasi

Gambar 5. 14 Hasil perhitungan kejelasan informasi yang diberikan ketika terjadi eror



Gambar 5. 15 Hasil perhitungan kesesuaian ukuran, bentuk, dan fungsi tombol pada aplikasi



Gambar 5. 16 Hasil perhitungan kesesuaian ukuran, bentuk, dan fungsi tombol pada aplikasi

5.3.3 Pengujian menggunakan perbandingan dengan tools Eclipse Metrics Plugin pada Cyclomatic Complexity

Pada program dilakukan juga uji coba perbandingan nilai untuk Cyclomatic Complexity dengan menggunakan Eclipse Metrics Plugin. Adapun hasi lyang didapatkan untuk 4 skenario yang berbeda adalah sebagai berikut :

Tabel 5. 12 Perbandingan Nilai CC memakai MetrycAnalyzer dengan Eclipse Metrics Plugin

Nama Program	Nilai CC	
	Eclipse Metrics Plugin	MetrycAnalyzer
Employments	7	7
TronGames	84	84
TicTacToe	72	72
OSM	328	328

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini akan diberikan kesimpulan yang diambil selama pengerjaan tugas akhir serta saran-saran tentang pengembangan yang dapat dilakukan terhadap tugas akhir ini di masa yang akan datang.

6.1 Kesimpulan

Dari hasil selama proses perancangan, implementasi, serta pengujian dapat diambil kesimpulan sebagai berikut:

1. Sistem dapat mendeteksi Ukuran Kode (Size) yang meliputi perhitungan semua fisik baris kode, Persentasi Komentar, Weighted Method Class (WMC), Response For Class (RFC), Number of Subunits, Coupling Between Object Classes (CBO), Depth of Inheritance Tree (DIT), dan Number of Children (NOC). Semua metrik tersebut yang tergolong *unused method* dapat ditangkap oleh sistem.
2. Berdasarkan hasil kuesioner yang telah dilakukan penulis, visualisasi atau tampilan dari aplikasi cukup mudah dimengerti.
3. Setiap metrik dapat dihitung dengan membaca setiap baris pada kode dan melakukan pencarian keywords untuk masing-masing metrik kemudian menampung hasil perhitungannya untuk diambil nilai akhir serta klasifikasinya.
4. Kode yang ada dalam file berhasil dideteksi dengan memindai keberadaan file lalu mengambil isinya dan ditampung dalam bentuk string.

6.2 Saran

Berikut saran-saran untuk pengembangan dan perbaikan sistem di masa yang akan datang. Diantaranya adalah sebagai berikut:

1. Menambahkan metrik-metrik lain yang belum ada di dalam aplikasi ini.
2. Menambahkan fitur untuk melakukan proses pengujian dalam bahasa lain

DAFTAR PUSTAKA

[1] M. H. a. B. Montazeri, "IEEE TRANSACTIONS ON SOFTWARE ENGINEERING," *Chidamber and Kemerer's Metrics Suite: A Measurment Theory Perspective,* vol. 22, April 1996.

[2] F. B. e. Abreu, "ECOOP'95," *Design Metrics for Object-Oriented Software Systems,* 1995.

[3] S. Chawla, "International Journal of Advanced Research in Computer Science and Software Engineering," *Review of MOOD and QMOOD metric sets,* vol. 3, no. 3, March 2013.

[4] K. B. J. V. B. Vinay Singh, "International Journal of Advanced Research in Computer Science and Software Engineering," *Analytical Evaluation of Class Complexity Metric,* vol. 3, no. 6, 2013.

[5] U. T. Ural Erdemir, "IEEE," *E-Quality: A Graph Based Object Oriented Software Quality Visualization Tool,* 2011.

[6] R. M. Michele Lanza, Object-Oriented Metrics in Practice: Using Software Metrics to Characterize, Evaluate, and Improve the Design of Object-Oriented Systems, Germany: Springer, 2006.

[7] T. J. McCABE, "IEEE TRANSACTIONS ON SOFTWARE ENGINEERING," *A Complexity Measure,* vol. 2, p. 308, December 1976.

[8] U. T. Ural Erdemir, "IEEE," *E-Quality : A Graph Based Object Oriented Software Quality Visualization Tool,* 2011.

[9] R. Arti Chhikara, "IJCSI International Journal of Computer Science Issues," *nalyzing the Complexity of Java Programs using Object -Oriented Software Metrics,* vol. 9, no. 1, January 2012.

[10] A. C. a. R.S.Chhillar, "IJCSI International Journal of

Computer Science Issues," *Analyzing the Complexity of Java Programs using Object - Oriented Software Metrics,* vol. 9, no. 1, 2012.

[11] V. B. KUMAR RAJNISH, "Class Inheritance Metrics-An Analytical and Empirical Approach," September 2007.

[12] "Eclipse," The eclipse foundation, [Online]. Available: https://eclipse.org/. [Diakses May 2015].

[13] "NetBeans IDE," [Online]. Available: https://netbeans.org/. [Diakses 2015 May 16].

[14] V. R. Basili, Fellow, IEEE, L. C. Briand dan W. L. Melo, "IEEE Transaction Software Engineering," *A Validation of Object-Oriented Design Metrics as Quality Indicators,* vol. 22, no. 10, 1996.

[15] L. A. A. Aloysius, "I.J. Information Technology and Computer Science," *Coupling Complexity Metric: A Cognitive Approach,* 2012.

[16] D. R. V. Vasudha Dixit, "International Journal of Engineering Research," *Static and Dynamic Coupling and Cohesion Measures in Object Oriented Programming,* vol. 2, no. 7, 2013.

[17] P. D. R. M. P. G. P. Lokesh Parasher, "International Journal of Research in IT, Management and Engineering," *A SURVEY OF OBJECT ORIENTED COUPLING METRICS,* vol. 1, no. 3.

[18] "wikipedia.com," wikipedia, [Online]. Available: http://en.wikipedia.org/wiki/Eclipse_(software). [Diakses 2015 May 20].

LAMPIRAN A : KODE SUMBER

Lampiran A berisi data kode yang digunakan sebagai skenario pengujian untuk MetrycAnalyzer. Yang pertama adalah kode GeneralTest1 yang merupakan program sederhana untuk mendata karyawan di suatu tempat. Yang kedua adalah kode program Tron. Untuk skenario ketiga, digunakan kode program TicTacToe, yaitu sebuah permainan layar 2dimensi. Skenario pengujian terakhir adalah kode program OSM. Kode-kode program ini menghasilkan nilai yang berbeda-beda untuk hasil penghitungannya. Berikut adalah kode-kode yang digunakan untuk skenario uji MetrycAnalyzer :

```
package metrycanalyzer;

import java.io.*;
public class employee
//employee
{
  DataInputStream in=new
  DataInputStream(System.in);
  private String name;
  //employee name
  public int number;
  //employee number
  public void getinfo(){
    System.out.println("Enter name:");
    name= in.readLine();
    System.out.println ("enter number :") ;
    number=Integer.parseInt(in.readLine()) ;
  }
  public void putinfo(){
    System.out.println("The name is:" +name);
    System.out.println ("Number=" +number);
  }
  public void show(){
    System.out.println("End of Employee ");
  }
}

public class generalmanager extends employee
//generalmanager
{
  private String title ;
```

```
  private double dues ;
  private int count;
  count = total;
  public void getinfo(){
    super.getinfo();
    System.out.println(”enter title :”);
    title=in.readLine();
    Console.WriteLine(“enter golf club dues:”);
    dues=double.parseDouble(in.readLine());
  }
  public void putinfo(){
    super.putinfo();
    System.out.println(count);
    System.out.println (“title:” +title);
    System.out.println(“dues:”+dues);
  }
  public void show1(){
    System.out.println(“End of manager class”);
  }
}

public class engineer extends employee
// engineer
{
  private int pubs ;
  public void getinfo(){
    super.getdata();
    System.out.println (”enter number of pubs:”) ;
    pubs=Integer.parseInt(in.readLine());
    }
  public void putinfo(){
    super.putinfo();
    System.out.println (“number of pubs:” +pubs);
  }
}

public class worker extends employee
// worker
{
  private int a;
  public int hours;
  public void getinfo(){
    super.getdata();
    System.out.println (“Enter number of hours:”) ;
    hours=Integer.parseInt(in.readLine());
  }
  public void calculate( ){
    int total=0;
    total = LEN*40;
```

```
  }
  public void putinfo(){
    super.putinfo();
    System.out.println ("number of hours :" +hours) ;
    System.out.println ("Total:" +total);
  }
}

public class hourlyemployee extends worker
//hourlyemployee
{
  private double sal;
  public void getinfo(){
    super.getinfo();
    System.out.println ("enter number of hours:") ;
    hours=Integer.parseInt(in.readLine());
  }
  public void salary(){
    sal=super.hours*250;
    // calling superclass instance variable
  }
  public void putinfo(){
    super.putinfo();
    System.out.println ("The salary is: " +sal);
  }
  public static void main(String args[])
    //main method
  {
    generalmanager m1 = new generalmanager();
    generalmanager m2 = new generalmanager();
    technician s1= new scientist();
    worker L1 = new worker();
    hourlyemployee h1 = new hourlyemployee();
    System.out.println ("Enter data for manager 1");
    //get data for several employees
    m1.getinfo();
    System.out.println ("Enter data for manager 2");
    m2.getinfo();
    System.out.println ("Enter data for scientist 1");
    s1.getinfo();
    System.out.println ("Enter data for laborer 1");
    L1.getinfo();
    System.out.println ("Enter data for hourlyemployee 1");
    h1.getinfo();
    System.out.println ("Data on manager 1");
                          m1.putinfo();
```

```
    System.out.println ("Data on manager 2 ");
    m2.putinfo();
```

```
    System.out.println ("Data on scientist 1");
    s1.putinfo();
    System.out.println ("Data on Laborer 1");
    L1.putinfo();
    System.out.println ("Data on hourly employee");
    h1.putinfo();
  }
}
```

Kode Lampiran 1 Kode GeneralTest1 untuk Pengujian

```
import java.awt.*;
import java.awt.BasicStroke;
import java.awt.Color;
import java.awt.Graphics2D;
import java.awt.Stroke;
import java.awt.Window;
import java.awt.Image;
import java.awt.image.BufferedImage;
import java.awt.image.BufferStrategy;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.awt.event.MouseMotionListener;
import java.util.ArrayList;
import javax.swing.JFrame;

public class Tron extends Core implements KeyListener,
MouseListener, MouseMotionListener {
  int centrex1 = 40;
  int centrey1 = 40;
  int centrex2 = 600;
  int centrey2 = 440;
  int currentDirection1 = 1;
  int currentDirection2 = 3;
  int moveAmount = 5;
  ArrayList<Integer> pathx1 = new ArrayList();
  ArrayList<Integer> pathy1 = new ArrayList();
  ArrayList<Integer> pathx2 = new ArrayList();
  ArrayList<Integer> pathy2 = new ArrayList();

  public void init() {
    super.init();
    Window w = sm.getFullScreenWindow();
    w.addKeyListener(this);
    w.addMouseListener(this);
    w.addMouseMotionListener(this);
  }
```

```
  public static void main(String[] args) {
    new Tron().run();
  }

  public void draw(Graphics2D g) {
    switch(currentDirection1){
      case 0:
        if (centrey1>0){
          centrey1-=moveAmount;
        } else {
          centrey1 = sm.getHeight();
        }
        break;
      case 1:
        if (centrex1 < sm.getWidth()){
          centrex1+=moveAmount;
        } else {
          centrex1 = 0;
        }
        break;
      case 2:
        if (centrey1 < sm.getHeight()){
          centrey1+=moveAmount;
        } else {
          centrey1 = 0;
        }
        break;
      case 3:
        if (centrex1>0){
          centrex1-=moveAmount;
        } else {
          centrex1 = sm.getWidth();
        }
        break;
    }
    switch(currentDirection2){
      case 0:
        if (centrey2>0){
          centrey2-=moveAmount;
        } else {
          centrey2 = sm.getHeight();
        }
        break;
case 1:
        if (centrex2 < sm.getWidth()){
          centrex2+=moveAmount;
        } else {
          centrex2 = 0;
```

```
        }
          break;
        case 2:
          if (centrey2 < sm.getHeight()){
            centrey2+=moveAmount;
          } else {
            centrey2 = 0;
          }
          break;
        case 3:
          if (centrex2>0){
            centrex2-=moveAmount;
          } else {
            centrex2 = sm.getWidth();
          }
          break;
      }
      for (int x = 0;x<pathx1.size();x++){
        if (((centrex1 == pathx1.get(x)) && (centrey1 ==
pathy1.get(x))) || ((centrex2 == pathx2.get(x)) && (centrey2
== pathy2.get(x))) || ((centrex1 == pathx2.get(x)) &&
(centrey1 == pathy2.get(x))) || ((centrex2 == pathx1.get(x))
&& (centrey2 == pathy1.get(x)))){
          System.exit(0);
        }
      }

      pathx1.add(centrex1);
      pathy1.add(centrey1);
      pathx2.add(centrex2);
      pathy2.add(centrey2);
      g.setColor(Color.BLACK);
      g.fillRect(0, 0, sm.getWidth(), sm.getHeight());

      for (int x = 0;x<pathx1.size();x++){
        g.setColor(Color.green);
        g.fillRect(pathx1.get(x), pathy1.get(x), 10, 10);
        g.setColor(Color.red);
        g.fillRect(pathx2.get(x), pathy2.get(x), 10, 10);
      }
    }
    public void keyPressed(KeyEvent e) {
      if (e.getKeyCode() == KeyEvent.VK_UP) {
        if (currentDirection1 != 2){
          currentDirection1 = 0;
        }
      } else if (e.getKeyCode() == KeyEvent.VK_DOWN) {
        if (currentDirection1 != 0){
          currentDirection1 = 2;
```

```
        }
      } else if (e.getKeyCode() == KeyEvent.VK_RIGHT) {
        if (currentDirection1 != 3){
          currentDirection1 = 1;
        }
      } else if (e.getKeyCode() == KeyEvent.VK_LEFT) {
        if (currentDirection1 != 1){
          currentDirection1 = 3;
        }
      }
      if (e.getKeyCode() == KeyEvent.VK_W){
        if (currentDirection2 != 2){
          currentDirection2 = 0;
        }
      } else if (e.getKeyCode() == KeyEvent.VK_S) {
        if (currentDirection2 != 0){
          currentDirection2 = 2;
        }
      } else if (e.getKeyCode() == KeyEvent.VK_D) {
        if (currentDirection2 != 3){
          currentDirection2 = 1;
        }
      } else if (e.getKeyCode() == KeyEvent.VK_A) {
        if (currentDirection2 != 1){
          currentDirection2 = 3;
        }
      }
    }

    public void keyReleased(KeyEvent e) {
    }

    public void keyTyped(KeyEvent arg0) {
    }

    public void mouseClicked(MouseEvent e) {
    }

    public void mouseEntered(MouseEvent arg0) {
}

    public void mouseExited(MouseEvent arg0) {
    }

    public void mousePressed(MouseEvent e) {
    }

    public void mouseReleased(MouseEvent e) {
    }
```

```
  public void mouseDragged(MouseEvent e) {
  }

  public void mouseMoved(MouseEvent e) {
  }
}

public abstract class Core {
  private static final DisplayMode modes[] = {
    //new DisplayMode(1920,1080,32,0),
    new DisplayMode(1680,1050,32,0),
    //new DisplayMode(1280,1024,32,0),
    new DisplayMode(800,600,32,0),
    new DisplayMode(800,600,24,0),
    new DisplayMode(800,600,16,0),
    new DisplayMode(640,480,32,0),
    new DisplayMode(640,480,24,0),
    new DisplayMode(640,480,16,0),
  };
  private boolean running;
  protected ScreenManager sm;

  public void stop(){
    running = false;
  }

  public void run(){
    try{
      init();
      gameLoop();
    }finally{
      sm.restoreScreen();
    }
  }

  public void init(){
    sm = new ScreenManager();
    DisplayMode dm = sm.findFirstCompatibaleMode(modes);
    sm.setFullScreen(dm);
    Window w = sm.getFullScreenWindow();
    w.setFont(new Font("Arial",Font.PLAIN,20));
    w.setBackground(Color.WHITE);
    w.setForeground(Color.RED);
    w.setCursor(w.getToolkit().createCustomCursor(new
BufferedImage(3, 3, BufferedImage.TYPE_INT_ARGB), new
Point(0, 0),"null"));
    running = true;
  }
```

```
  public void gameLoop(){
    long startTime = System.currentTimeMillis();
    long cumTime = startTime;

    while (running){
      long timePassed = System.currentTimeMillis()-cumTime;
      cumTime+= timePassed;
      update(timePassed);
      Graphics2D g = sm.getGraphics();
      draw(g);
      g.dispose();
      sm.update();

      try{
        Thread.sleep(20);
      }catch(Exception ex){}
    }
  }

  public void update(long timePassed){}
  public abstract void draw(Graphics2D g);
}

public class ScreenManager {
  private GraphicsDevice vc;
  public ScreenManager(){
    GraphicsEnvironment e =
GraphicsEnvironment.getLocalGraphicsEnvironment();
    vc = e.getDefaultScreenDevice();
  }

  public DisplayMode[] getCompatibleDisplayModes(){
    return vc.getDisplayModes();
  }

   public DisplayMode findFirstCompatibaleMode(DisplayMode[]
modes){
    DisplayMode goodModes[] = vc.getDisplayModes();
    for(int x = 0; x<modes.length;x++){
      for(int y = 0;y<goodModes.length;y++){
        if(displayModesMatch(modes[x],goodModes[y])){
          return modes[x];
        }
      }
    }
    return null;
  }
```

```
  public DisplayMode getCurrentDM(){
    return vc.getDisplayMode();
  }

  public boolean displayModesMatch(DisplayMode m1,
DisplayMode m2){
    if(m1.getWidth() != m2.getWidth() || m1.getHeight() !=
m2.getHeight()){
      return false;
    }
    if(m1.getBitDepth() != DisplayMode.BIT_DEPTH_MULTI &&
m2.getBitDepth() != DisplayMode.BIT_DEPTH_MULTI &&
m1.getBitDepth() != m2.getBitDepth()){
    return false;
    }

    if(m1.getRefreshRate() !=
DisplayMode.REFRESH_RATE_UNKNOWN && m2.getRefreshRate() !=
DisplayMode.REFRESH_RATE_UNKNOWN && m1.getRefreshRate() !=
m2.getRefreshRate()){
      return false;
    }
    return true;
  }

  public void setFullScreen(DisplayMode dm){
    JFrame f = new JFrame();
    f.setUndecorated(true);
    f.setIgnoreRepaint(true);
    f.setResizable(false);
    vc.setFullScreenWindow(f);

    if(dm != null && vc.isDisplayChangeSupported()){
      try{
        vc.setDisplayMode(dm);
      }catch(Exception ex){}
f.createBufferStrategy(2);
      }
    }
  }
  public Graphics2D getGraphics(){
    Window w = vc.getFullScreenWindow();
    if(w != null){
      BufferStrategy bs = w.getBufferStrategy();
      return (Graphics2D)bs.getDrawGraphics();
    }else{
      return null;
    }
  }
```

```
  public void update(){
    Window w = vc.getFullScreenWindow();
    if(w != null){
      BufferStrategy bs = w.getBufferStrategy();
      if(!bs.contentsLost()){
        bs.show();
      }
    }
  }

  public Window getFullScreenWindow(){
    return vc.getFullScreenWindow();
  }


  public int getWidth(){
    Window w = vc.getFullScreenWindow();
    if(w != null){
      return w.getWidth();
    }else{
      return 0;
    }
  }

  public int getHeight(){
    Window w = vc.getFullScreenWindow();
    if(w != null){
      return w.getHeight();
    }else{
      return 0;
    }
  }

  public void restoreScreen(){
    Window w = vc.getFullScreenWindow();
    if(w != null){
      w.dispose();
    }
    vc.setFullScreenWindow(null);
  }

  public BufferedImage createCompatibaleimage(int w, int h,
int t){
    Window win = vc.getFullScreenWindow();
    if(win != null){
      GraphicsConfiguration gc =
win.getGraphicsConfiguration();
      return gc.createCompatibleImage(w,h,t);
```

```
    }else{
      return null;
    }
  }
}

public class Animation {
  private ArrayList scenes;
  private int sceneIndex;
  private long movieTime;
  private long totalTime;

  public Animation(){
    scenes = new ArrayList();
    totalTime = 0;
    start();
  }

  public synchronized void addScene(Image i, long t){
    totalTime += t;
    scenes.add(new oneScene(i,totalTime));
  }

  public synchronized void start(){
    movieTime = 0;
    sceneIndex = 0;
  }

  public synchronized void update(long timePassed){
    if(scenes.size()>1){
      movieTime += timePassed;
      if(movieTime>=totalTime){
        movieTime = 0;
        sceneIndex = 0;
      }
      while(movieTime > getScene(sceneIndex).endTime){
        sceneIndex++;
      }
    }
  }

  public synchronized Image getImage(){
    if(scenes.size()==0){
      return null;
    }else{
      return getScene(sceneIndex).pic;
    }
  }
```

```
  private oneScene getScene(int x){
    return (oneScene)scenes.get(x);
  }

  private class oneScene{
    Image pic;
    long endTime;
    public oneScene(Image pic,long endTime){
      this.pic = pic;
      this.endTime = endTime;
    }
  }
}
```

Kode Lampiran 2 Kode Tron untuk Pengujian

```
import java.util.logging.Level;
import java.util.logging.Logger;
import java.io.*;
import java.util.ArrayList;

public class TicTacToe extends javax.swing.JFrame {
  static int winComb[][] =
{{1,2,3},{4,5,6},{7,8,9},{1,4,7},{2,5,8},{3,6,9},{1,5,9},{3,
5,7}};
  public static int[][] state = {{0,0,0},{0,0,0},{0,0,0}};
  Player pl1 = new Human();
  Player pl2 = new Computer("mind\\layer");
  public static int butClicked = 0;
  int w1=0 , w2 = 0 , dr = 0;
  public TicTacToe() {
    initComponents();
  }
public void start(){
    if(w1==500)System.exit(0);
    int current = 1 , turn = 1;
    int w=0;
    while((w=checkWin(turn,state))==0){
      if(current == 1){
        pl1.playTurn(1,turn);
        refreshGrid();
        current = 2;
      }else if(current == 2 ){
        pl2.playTurn(2,turn);
        refreshGrid();
```

```
        current = 1;
```

```
      }
      turn++;
      try{
        Thread.sleep(0);
      } catch (InterruptedException ex){
        Logger.getLogger
(TicTacToe.class.getName()).log(Level.SEVERE, null, ex);
      }
    }
    if(w==1) {
      pl1.notifyWin(1);
      pl2.notifyLose(2);
      print("Player 1 Won The Game !");
      w1++;
    }else if(w==2) {
      pl2.notifyWin(1);
      pl1.notifyLose(2);
      print("Player 2 Won The Game !");
      w2++;
    }else if(w==-1) {
      print("Game DRAW !");
      dr++;
    }
    try {
      Thread.sleep(0);
    } catch (InterruptedException ex){
      Logger.getLogger
(TicTacToe.class.getName()).log(Level.SEVERE, null, ex);
    }
  }

  public void refreshGrid(){
    b11.setText(state[0][0]==1?"X":(state[0][0]==2?"O":""));
b12.setText(state[0][1]==1?"X":(state[0][1]==2?"O":""));
    b13.setText(state[0][2]==1?"X":(state[0][2]==2?"O":""));
    b21.setText(state[1][0]==1?"X":(state[1][0]==2?"O":""));
    b22.setText(state[1][1]==1?"X":(state[1][1]==2?"O":""));
    b23.setText(state[1][2]==1?"X":(state[1][2]==2?"O":""));
    b31.setText(state[2][0]==1?"X":(state[2][0]==2?"O":""));
    b32.setText(state[2][1]==1?"X":(state[2][1]==2?"O":""));
    b33.setText(state[2][2]==1?"X":(state[2][2]==2?"O":""));
    jLabel1.setText(" X wins : "+w1);
    jLabel2.setText(" O wins : "+w2);
    jLabel3.setText(" Draws  : "+dr);
  }
  public static int checkWin(int turn,int[][] st){
    int ret = 0;
    String x ="";
    String o ="";
```

```
    int i=0 , j=0 , c=0 , p , q;
    for(p=0;p<3;p++){
      for(q=0;q<3;q++){
        c++;
        if(st[p][q]==1){
          x+=c;
        }else if(st[p][q]==2){
          o+=c;
        }
      }
    }
    //print(x+" : "+o);
    ret = checkWin2(x,o);
    if(turn==10 && ret==0){
      ret = -1;
    }
    return ret;
  }
  public static int checkWin2(String x,String o){
    int ret = 0;
    int p;
    for(p=0;p<8;p++){
      if(x.indexOf((char)winComb[p][0]+'0')>-1 &&
x.indexOf((char)winComb[p][1]+'0')>-1 &&
x.indexOf((char)winComb[p][2]+'0')>-1){
        ret = 1;
        break;
      }
      if(o.indexOf((char)winComb[p][0]+'0')>-1 &&
o.indexOf((char)winComb[p][1]+'0')>-1 &&
o.indexOf((char)winComb[p][2]+'0')>-1) {
        ret = 2;
        break;
      }
    }
    return ret;
  }
  public void print(String s){
    Notification.setText("\t"+s);
  }

  public void gameInit(){
    state[0][0] = 0;
    state[0][1] = 0;
    state[0][2] = 0;
    state[1][0] = 0;
    state[1][1] = 0;
    state[1][2] = 0;
    state[2][0] = 0;
```

```
    state[2][1] = 0;
    state[2][2] = 0;
    refreshGrid();
  }

  @SuppressWarnings("unchecked")
  // <editor-fold defaultstate="collapsed" desc="Generated
Code">//GEN-BEGIN:initComponents
  private void initComponents() {
    try {
      jPanel1
=(javax.swing.JPanel)java.beans.Beans.instantiate(getClass()
.getClassLoader(), "TicTacToe_jPanel1");
    } catch (ClassNotFoundException e) {
      e.printStackTrace();
    } catch (java.io.IOException e) {
      e.printStackTrace();
    }
    b21 = new javax.swing.JButton();
    b11 = new javax.swing.JButton();
    b22 = new javax.swing.JButton();
    b12 = new javax.swing.JButton();
    b13 = new javax.swing.JButton();
    b23 = new javax.swing.JButton();
    b31 = new javax.swing.JButton();
    b32 = new javax.swing.JButton();
    b33 = new javax.swing.JButton();
          Notification = new javax.swing.JLabel();
    jPanel2 = new javax.swing.JPanel();
    jLabel1 = new javax.swing.JLabel();
    jLabel2 = new javax.swing.JLabel();
    jLabel3 = new javax.swing.JLabel();
    jLabel4 = new javax.swing.JLabel();
    jLabel5 = new javax.swing.JLabel();
    jLabel6 = new javax.swing.JLabel();
    setDefaultCloseOperation
(javax.swing.WindowConstants.EXIT_ON_CLOSE);
    setAlwaysOnTop(true);
    setPreferredSize(new java.awt.Dimension(600, 400));
    b21.setBackground(new java.awt.Color(255, 255, 255));
    b21.setFont(new java.awt.Font("Arial", 1, 48)); //
NOI18N
    b21.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
    b21.addActionListener(new
java.awt.event.ActionListener(){
     public void actionPerformed(java.awt.event.ActionEvent
evt) {
       b21ActionPerformed(evt);}
```

```
    });
    b11.setBackground(new java.awt.Color(255, 255, 255));
    b11.setFont(new java.awt.Font("Arial", 1, 48)); //
NOI18N
    b11.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
    b11.addActionListener(new
java.awt.event.ActionListener(){
      public void actionPerformed(java.awt.event.ActionEvent
evt) { b11ActionPerformed(evt);}
    });
    b22.setBackground(new java.awt.Color(255, 255, 255));
    b22.setFont(new java.awt.Font("Arial", 1, 48)); //
NOI18N
    b22.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
    b22.addActionListener(new
java.awt.event.ActionListener(){
      public void actionPerformed(java.awt.event.ActionEvent
evt) {
        b22ActionPerformed(evt);}
    });
    b12.setBackground(new java.awt.Color(255, 255, 255));
    b12.setFont(new java.awt.Font("Arial", 1, 48)); //
NOI18N
                        b12.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
    b12.addActionListener(new
java.awt.event.ActionListener(){
      public void actionPerformed(java.awt.event.ActionEvent
evt) {
        b12ActionPerformed(evt);
      }
    });
    b13.setBackground(new java.awt.Color(255, 255, 255));
    b13.setFont(new java.awt.Font("Arial", 1, 48)); //
NOI18N
    b13.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
    b13.addActionListener(new
java.awt.event.ActionListener(){
      public void actionPerformed(java.awt.event.ActionEvent
evt) {
        b13ActionPerformed(evt);
      }
    });
    b23.setBackground(new java.awt.Color(255, 255, 255));
    b23.setFont(new java.awt.Font("Arial", 1, 48)); //
NOI18N
```

```
    b23.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
    b23.addActionListener(new
java.awt.event.ActionListener(){
      public void actionPerformed(java.awt.event.ActionEvent
evt) {
        b23ActionPerformed(evt);
      }
    });
    b31.setBackground(new java.awt.Color(255, 255, 255));
    b31.setFont(new java.awt.Font("Arial", 1, 48)); //
NOI18N
    b31.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
    b31.addActionListener(new
java.awt.event.ActionListener(){
      public void actionPerformed(java.awt.event.ActionEvent
evt) {
        b31ActionPerformed(evt);
        }
      });
    b32.setBackground(new java.awt.Color(255, 255, 255));
    b32.setFont(new java.awt.Font("Arial", 1, 48)); //
NOI18N
    b32.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
    b32.addActionListener(new
java.awt.event.ActionListener(){
      public void actionPerformed(java.awt.event.ActionEvent
evt) {

 b32ActionPerformed(evt);
      }
    });

    b33.setBackground(new java.awt.Color(255, 255, 255));
    b33.setFont(new java.awt.Font("Arial", 1, 48)); //
NOI18N
    b33.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
    b33.addActionListener(new
java.awt.event.ActionListener(){
      public void actionPerformed(java.awt.event.ActionEvent
evt) {
        b33ActionPerformed(evt);
      }
    });
```

```
    Notification.setBackground(new java.awt.Color(255, 255,
0));
    Notification.setFont(new java.awt.Font("Tahoma", 0,
18)); // NOI18N
    Notification.setForeground(new java.awt.Color(0, 0,
102));
    Notification.setText("Tic - Tac - Toe");
    Notification.setBorder(new
javax.swing.border.MatteBorder(null));

    jLabel1.setFont(new java.awt.Font("Arial", 0, 18));
//NOI18N
    jLabel1.setBorder
(javax.swing.BorderFactory.createLineBorder (new
java.awt.Color(0, 0, 0)));

    jLabel2.setFont(new java.awt.Font("Arial",0,18));//
NOI18N
    jLabel2.setBorder
(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));
    jLabel3.setFont(new java.awt.Font("Arial",0,18));//
NOI18N
    jLabel3.setBorder
(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));

    jLabel4.setFont(new java.awt.Font("Arial",0,18));//
NOI18N
    jLabel4.setBorder
(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));

    jLabel5.setFont(new java.awt.Font("Arial",0,18));//
NOI18N
    jLabel5.setBorder
(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));

    jLabel6.setFont(new java.awt.Font("Arial",0,18));//
NOI18N
    jLabel6.setBorder
(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));

    javax.swing.GroupLayout jPanel2Layout = new
javax.swing.GroupLayout(jPanel2);
    jPanel2.setLayout(jPanel2Layout);
    jPanel2Layout.setHorizontalGroup(
```

```
    jPanel2Layout.createParallelGroup
(javax.swing.GroupLayout.Alignment.LEADING)
      .addGroup(jPanel2Layout.createSequentialGroup()
      .addContainerGap()
      .addGroup(jPanel2Layout.createParallelGroup
(javax.swing.GroupLayout.Alignment.LEADING)
      .addComponent(jLabel1,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
      .addComponent(jLabel2,
javax.swing.GroupLayout.DEFAULT_SIZE, 203, Short.MAX_VALUE)
      .addComponent(jLabel3,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
      .addComponent(jLabel4,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .addComponent(jLabel5,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
      .addComponent(jLabel6,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
      .addContainerGap())
    );
    jPanel2Layout.setVerticalGroup(
    jPanel2Layout.createParallelGroup
(javax.swing.GroupLayout.Alignment.LEADING)
      .addGroup(jPanel2Layout.createSequentialGroup()
      .addContainerGap()
      .addComponent(jLabel1,
javax.swing.GroupLayout.PREFERRED_SIZE, 40,
javax.swing.GroupLayout.PREFERRED_SIZE)
      .addPreferredGap
(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
      .addComponent(jLabel2,
javax.swing.GroupLayout.PREFERRED_SIZE, 40,
javax.swing.GroupLayout.PREFERRED_SIZE)
      .addPreferredGap
(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
      .addComponent(jLabel3,
javax.swing.GroupLayout.PREFERRED_SIZE, 40,
javax.swing.GroupLayout.PREFERRED_SIZE)
      .addPreferredGap
(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
      .addComponent(jLabel4,
javax.swing.GroupLayout.PREFERRED_SIZE, 40,
javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
      .addPreferredGap
(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
      .addComponent(jLabel5,
javax.swing.GroupLayout.PREFERRED_SIZE, 40,
javax.swing.GroupLayout.PREFERRED_SIZE)
      .addPreferredGap
(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
      .addComponent(jLabel6,
javax.swing.GroupLayout.PREFERRED_SIZE, 40,
javax.swing.GroupLayout.PREFERRED_SIZE)
      .addContainerGap
(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
    );
    jLabel1.getAccessibleContext().setAccessibleName("l1");
    jLabel1.getAccessibleContext().setAccessibleDescription
("");
javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
    jPanel1.setLayout(jPanel1Layout);
    jPanel1Layout.setHorizontalGroup(
      jPanel1Layout.createParallelGroup
(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel1Layout.createSequentialGroup()
        .addContainerGap()
        .addGroup(jPanel1Layout.createParallelGroup
(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(Notification,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addGroup(jPanel1Layout.createSequentialGroup()
        .addGroup(jPanel1Layout.createParallelGroup
(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel1Layout.createSequentialGroup()
        .addGroup(jPanel1Layout.createParallelGroup
(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(b21,
javax.swing.GroupLayout.PREFERRED_SIZE, 100,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(b11,
javax.swing.GroupLayout.PREFERRED_SIZE, 100,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addPreferredGap
(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(jPanel1Layout.createParallelGroup
(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel1Layout.createSequentialGroup()
        .addComponent(b22,
javax.swing.GroupLayout.PREFERRED_SIZE, 100,
javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
        .addPreferredGap
(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(b23,
javax.swing.GroupLayout.PREFERRED_SIZE, 100,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGroup(jPanel1Layout.createSequentialGroup()
        .addComponent(b12,
javax.swing.GroupLayout.PREFERRED_SIZE, 100,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addPreferredGap
(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(b13,
javax.swing.GroupLayout.PREFERRED_SIZE, 100,
javax.swing.GroupLayout.PREFERRED_SIZE))))
        .addGroup(jPanel1Layout.createSequentialGroup()
        .addComponent(b31,
javax.swing.GroupLayout.PREFERRED_SIZE, 100,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addPreferredGap
(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(b32,
javax.swing.GroupLayout.PREFERRED_SIZE, 100,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addPreferredGap
(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(b33,
javax.swing.GroupLayout.PREFERRED_SIZE, 100,
javax.swing.GroupLayout.PREFERRED_SIZE)))
        .addPreferredGap
(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(jPanel2,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))
        .addContainerGap())
      );
    jPanel1Layout.setVerticalGroup(
    jPanel1Layout.createParallelGroup
(javax.swing.GroupLayout.Alignment.LEADING)
      .addGroup(jPanel1Layout.createSequentialGroup()
      .addContainerGap()
      .addGroup(jPanel1Layout.createParallelGroup
(javax.swing.GroupLayout.Alignment.LEADING, false)
      .addGroup(jPanel1Layout.createSequentialGroup()
      .addGroup(jPanel1Layout.createParallelGroup
(javax.swing.GroupLayout.Alignment.LEADING)
      .addComponent(b12,
javax.swing.GroupLayout.PREFERRED_SIZE, 100,
javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
      .addComponent(b13,
javax.swing.GroupLayout.PREFERRED_SIZE, 100,
javax.swing.GroupLayout.PREFERRED_SIZE)
      .addComponent(b11,
javax.swing.GroupLayout.PREFERRED_SIZE, 100,
javax.swing.GroupLayout.PREFERRED_SIZE))
      .addPreferredGap
(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
      .addGroup(jPanel1Layout.createParallelGroup
(javax.swing.GroupLayout.Alignment.BASELINE)
                        .addComponent(b22,
javax.swing.GroupLayout.PREFERRED_SIZE, 100,
javax.swing.GroupLayout.PREFERRED_SIZE)
      .addComponent(b21,
javax.swing.GroupLayout.PREFERRED_SIZE, 100,
javax.swing.GroupLayout.PREFERRED_SIZE)
      .addComponent(b23,
javax.swing.GroupLayout.PREFERRED_SIZE, 100,
javax.swing.GroupLayout.PREFERRED_SIZE))
      .addPreferredGap
(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
      .addGroup(jPanel1Layout.createParallelGroup
(javax.swing.GroupLayout.Alignment.BASELINE)
      .addComponent(b32,
javax.swing.GroupLayout.PREFERRED_SIZE, 100,
javax.swing.GroupLayout.PREFERRED_SIZE)
      .addComponent(b31,
javax.swing.GroupLayout.PREFERRED_SIZE, 100,
javax.swing.GroupLayout.PREFERRED_SIZE)
      .addComponent(b33,
javax.swing.GroupLayout.PREFERRED_SIZE, 100,
javax.swing.GroupLayout.PREFERRED_SIZE)))
      .addComponent(jPanel2,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
      .addPreferredGap
(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
      .addComponent(Notification,
javax.swing.GroupLayout.DEFAULT_SIZE, 40, Short.MAX_VALUE)
      .addContainerGap())
    );

    javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(
      layout.createParallelGroup
(javax.swing.GroupLayout.Alignment.LEADING)
      .addGroup(layout.createSequentialGroup()
```

```
      .addComponent(jPanel1,
javax.swing.GroupLayout.PREFERRED_SIZE, 561,
javax.swing.GroupLayout.PREFERRED_SIZE)
      .addGap(0, 0, Short.MAX_VALUE))
    );
    layout.setVerticalGroup(
                    layout.createParallelGroup
          (javax.swing.GroupLayout.Alignment.LEADING)

      .addComponent(jPanel1,
javax.swing.GroupLayout.DEFAULT_SIZE, 385, Short.MAX_VALUE)
    );

    pack();
  }// </editor-fold>//GEN-END:initComponents

  private void b33ActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_b33ActionPerformed
    if(state[2][2]==0)
      butClicked = 9;
    }//GEN-LAST:event_b33ActionPerformed
    private void
b32ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_b32ActionPerformed
      if(state[2][1]==0)
      butClicked = 8;
    }//GEN-LAST:event_b32ActionPerformed
    private void
b31ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_b31ActionPerformed
      if(state[2][0]==0)
      butClicked = 7;
    }//GEN-LAST:event_b31ActionPerformed
    private void
b23ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_b23ActionPerformed
      if(state[1][2]==0)
      butClicked = 6;
    }//GEN-LAST:event_b23ActionPerformed
    private void
b13ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_b13ActionPerformed
      if(state[0][2]==0)
      butClicked = 3;
    }//GEN-LAST:event_b13ActionPerformed
    private void
b12ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_b12ActionPerformed
      if(state[0][1]==0)
```

```
      butClicked = 2;
    }//GEN-LAST:event_b12ActionPerformed

    private void
b22ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_b22ActionPerformed
if(state[1][1]==0)
      butClicked = 5;
    }//GEN-LAST:event_b22ActionPerformed

    private void
b11ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_b11ActionPerformed
      if(state[0][0]==0)
      butClicked = 1;
    }//GEN-LAST:event_b11ActionPerformed
    private void
b21ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_b21ActionPerformed
      if(state[1][0]==0)
      butClicked = 4;
    }//GEN-LAST:event_b21ActionPerformed
    public static void main(String args[]) {
      /* Set the Nimbus look and feel */
      //<editor-fold defaultstate="collapsed" desc=" Look
and feel setting code (optional) ">
      /* If Nimbus (introduced in Java SE 6) is not
available, stay with the default look and feel.
      * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfe
el/plaf.html
      */
      try {
        for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
          if ("Nimbus".equals(info.getName())) {
            javax.swing.UIManager.setLookAndFeel
(info.getClassName());
            break;
          }
        }
      } catch (ClassNotFoundException ex) {
        java.util.logging.Logger.getLogger
(TicTacToe.class.getName()).log(java.util.logging.Level.SEVE
RE, null, ex);
      } catch (InstantiationException ex) {
        java.util.logging.Logger.getLogger
(TicTacToe.class.getName()).log(java.util.logging.Level.SEVE
RE, null, ex);
```

```
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger
(TicTacToe.class.getName()).log(java.util.logging.Level.SEVE
RE, null, ex);
} catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger
(TicTacToe.class.getName()).log(java.util.logging.Level.SEVE
RE, null, ex);
        }
      //</editor-fold>
      TicTacToe t = new TicTacToe();
      t.setVisible(true);
      while(true){
        t.start();
        t.gameInit();
      }
    }

    // Variables declaration - do not modify
    //GEN-BEGIN:variables
    private javax.swing.JLabel Notification;
    public static javax.swing.JButton b11;
    public static javax.swing.JButton b12;
    public static javax.swing.JButton b13;
    public static javax.swing.JButton b21;
    public static javax.swing.JButton b22;
    public static javax.swing.JButton b23;
    public static javax.swing.JButton b31;
    public static javax.swing.JButton b32;
    public static javax.swing.JButton b33;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    // End of variables declaration//GEN-END:variables
}

public abstract class Player {
    void playTurn(int pl,int turn){}
    void playerInit(){}
    void notifyWin(int pl){}
    void notifyLose(int pl){}
}
public class Human extends Player {
    @Override
```

```
  void playTurn(int pl,int turn){
    while(TicTacToe.butClicked == 0);
      switch(TicTacToe.butClicked){
        case 1 : TicTacToe.state[0][0]=pl;break;
        case 2 : TicTacToe.state[0][1]=pl;break;
        case 3 : TicTacToe.state[0][2]=pl;break;
        case 4 : TicTacToe.state[1][0]=pl;break;
        case 5 : TicTacToe.state[1][1]=pl;break;
        case 6 : TicTacToe.state[1][2]=pl;break;
        case 7 : TicTacToe.state[2][0]=pl;break;
        case 8 : TicTacToe.state[2][1]=pl;break;
        case 9 : TicTacToe.state[2][2]=pl;break;
      }
    TicTacToe.butClicked = 0;
  }
}

public class Computer extends Player {
  int t=0;
  Node begin = new Node("000000000",0,this);
  Node current = begin;
  double lr = 0;
  File[] layerFiles = new File[9];
  ArrayList<Layer> layers = new ArrayList<Layer>();
  public Computer(String l){
    for(int i=0;i<9;i++){
      layerFiles[i] = new File(l+(i+1)+".nodes");
      if(!layerFiles[i].exists()){
        try{
          layerFiles[i].createNewFile();
        }catch(Exception e) {
          System.out.println(e);
        }
      }
    }
    //loadMind();
    makeNewMind();evaluate();saveMind();
    begin.subNodes="100000000,010000000,001000000,000100000,
000010000,000001000,000000100,000000010,000000001,";
    begin.extractNodes();
  }
  public void evaluate(){
    for(int i=0;i<9;i++){
      Layer l = layers.get(i);
    for(int j=0;j<l.nodes.size();j++){
        String x="" , o="" , s = l.nodes.get(j).id;
        for(int k=0;k<9;k++){
          char ch = s.charAt(k);
          if(ch=='1') {
```

```
            x+=(k+1);
          }else if(ch=='2'){
            o+=(k+1);
          }
        }
        int r = TicTacToe.checkWin2(x,o);
        switch(r) {
          case 1 : l.nodes.get(j).pref=50;
            l.nodes.get(j).nodeType=1;
            break;
          case 2 : l.nodes.get(j).pref=-50;
            l.nodes.get(j).nodeType=1;
            break;
          case -1 :
            l.nodes.get(j).nodeType=1;
            break;
        }
      }
    }
  }
  public void loadMind(){
    FileReader f;
    BufferedReader r;
    int i=0;
    try{
      for(int l=0;l<9;l++){
        layers.add(new Layer(l+1));
        f = new FileReader(layerFiles[l]);
        r = new BufferedReader(f);
        String line;
        while((line=r.readLine())!=null) {
          Node temp = new Node(r.readLine(),l+1,this);
          temp.subNodes = r.readLine();
          String no = r.readLine();
          //System.out.println(no);
          temp.pref = Integer.parseInt(no);
          temp.n = Integer.parseInt(r.readLine());
          temp.nodeType = Integer.parseInt(r.readLine());
          layers.get(l).nodes.add(temp);
        }
      }
for(int l=0;l<9;l++){
        for(int j=0;j<layers.get(l).nodes.size();j++){
          layers.get(l).nodes.get(j).extractNodes();
        }
      }
    }
    catch(Exception e){
      e.printStackTrace(System.out);
```

```
    }
  }
  public void saveMind(){
    for(int i=7;i>=0;i--){layers.get(i).refreshLayer();
    }
    try{ for(int i=0;i<9;i++){
        Layer l = layers.get(i);
        PrintWriter p = new PrintWriter(new
BufferedWriter(new FileWriter(layerFiles[i])));
        for(int j=0;j<l.nodes.size();j++){
          Node temp = l.nodes.get(j);
          p.println("************************************");
          p.println(temp.id);
          //System.out.println(temp.id);
          String s="";
          for(int k=0;k<temp.sub_nodes.size();k++){
            s+=temp.sub_nodes.get(k).id+",";
          }
          p.println(s);
          p.println(temp.pref);
          p.println(temp.n);
          p.println(temp.nodeType);
        }
        p.close();
        //System.out.println("layer "+i+" :
"+l.nodes.size());
      }
    }catch(Exception e) {
      e.printStackTrace(System.out);
    }
  }
  public void makeNewMind(){
    layers.add(new Layer(1));
    layers.add(new Layer(2));
    layers.add(new Layer(3));
    layers.add(new Layer(4));
    layers.add(new Layer(5));
    layers.add(new Layer(6));
    layers.add(new Layer(7));
    layers.add(new Layer(8));
    layers.add(new Layer(9));

    for(int i1=0;i1<=2;i1++){
      for(int i2=0;i2<=2;i2++){
        for(int i3=0;i3<=2;i3++){
          for(int i4=0;i4<=2;i4++){
            for(int i5=0;i5<=2;i5++){
              for(int i6=0;i6<=2;i6++){
                for(int i7=0;i7<=2;i7++){
```

```
                    for(int i8=0;i8<=2;i8++){
                      for(int i9=0;i9<=2;i9++){
                        int l=9;
                        if(i1==0)l--;
                        if(i2==0)l--;
                        if(i3==0)l--;
                        if(i4==0)l--;
                        if(i5==0)l--;
                        if(i6==0)l--;
                        if(i7==0)l--;
                        if(i8==0)l--;
                        if(i9==0)l--;

                        int x=0;
                        if(i1==1)x++;
                        if(i2==1)x++;
                        if(i3==1)x++;
                        if(i4==1)x++;
                        if(i5==1)x++;
                        if(i6==1)x++;
                        if(i7==1)x++;
                        if(i8==1)x++;
                        if(i9==1)x++;

                        int o = l-x;
                        if((x-o==0 || x-o==1) && l!=0 ) {
                          String id =
""+i1+i2+i3+i4+i5+i6+i7+i8+i9;
                        layers.get(l-1).nodes.add(new
Node(id,l,this));
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    }
    for(int l=1;l<9;l++){
      for(int j=0;j<layers.get(l).nodes.size();j++){
        Node node = layers.get(l).nodes.get(j);
        for(int i=0;i<9;i++){
          String newId="";
          for(int k=0;k<9;k++){
            char ch = node.id.charAt(k);
            if(k==i)ch='0';
            newId+=ch;
```

```
          }
          if(!newId.equals(node.id)) {
            try{
              layers.get(l-
1).searchById(newId).sub_nodes.add(node);
              //System.out.println(node.id+" : "+newId+" :
"+l);
            }catch(NullPointerException e){}
          }
        }
      }
    }
    begin.extractNodes();
  }

  @Override
  public void playTurn(int p,int turn){
    t = turn;
    if(turn != 1) {
      current = layers.get(turn-
2).searchByState(TicTacToe.state);
    }
    if(turn == 1){
      current = begin;
    }
    if(p==1)current.playNextn();
    if(p==2)current.playNext2();
  }

  @Override
  void notifyWin(int pl){
    if(pl==1)current.pref+=10;
    if(pl==2)current.pref-=10;
    current.nodeType=1;
//System.out.println("*******"+current.id+":"+current.pref+"
: "+layers.get(current.lNum-1).searchById(current.id).pref);
    saveMind();
  }
  @Override
  void notifyLose(int pl){
    if(pl==1)current.pref-=10;
    if(pl==2)current.pref+=10;
    current.nodeType=1;

//System.out.println("*******"+current.id+":"+current.pref+"
: "+layers.get(current.lNum-1).searchById(current.id).pref);
    saveMind();
  }
}
```

```
public class Node {
  ArrayList<Node> sub_nodes = new ArrayList<Node>();
  String subNodes="";
  String id="";
  int pref = 0;
  int n=0;
  int lNum;
  Computer comp;
  int nodeType=0;
  public Node(String i,int l,Computer c){
    id=i;
    lNum = l;
    comp = c;
  }

  public void setAsState(){
    for(int i=0;i<id.length();i++){
      int val = (int)(id.charAt(i)-'0');
      int t = i/3;
      TicTacToe.state[t][i-(t*3)] = val;
    }
  }
  public void playNext1(){
    ArrayList<Node> temp = new ArrayList<Node>();
    //System.out.println(id+" : "+sub_nodes.size());
    long max = sub_nodes.get(0).pref;
    for(int i=0;i<sub_nodes.size();i++){
      if(sub_nodes.get(i).pref > max) {
        temp.clear();
        temp.add(sub_nodes.get(i));
        max = sub_nodes.get(i).pref;
}else if(sub_nodes.get(i).pref == max) {
        temp.add(sub_nodes.get(i));
      }
    }

    int choice = (int) (Math.random()*temp.size());
    temp.get(choice).n++;
    temp.get(choice).setAsState();
  }

  public void playNext2(){
    ArrayList<Node> temp = new ArrayList<Node>();
    //System.out.println(id+" : "+sub_nodes.size());
    long min = sub_nodes.get(0).pref;
    for(int i=0;i<sub_nodes.size();i++){
      if(sub_nodes.get(i).pref < min) {
        temp.clear();
```

```
        temp.add(sub_nodes.get(i));
        min = sub_nodes.get(i).pref;
      }else if(sub_nodes.get(i).pref == min) {
        temp.add(sub_nodes.get(i));
      }
    }

    int choice = (int) (Math.random()*temp.size());
    temp.get(choice).n++;
    temp.get(choice).setAsState();
  }

  public void playNextn(){
    ArrayList<Node> temp = new ArrayList<Node>();
    //System.out.println(id+" : "+sub_nodes.size());
    int choice = 0;
    long min = sub_nodes.get(0).n;
    for(int i=0;i<sub_nodes.size();i++){
      if(sub_nodes.get(i).n < min) {
        min = sub_nodes.get(i).n;
        choice = i;
      }
    }
    sub_nodes.get(choice).n++;
    sub_nodes.get(choice).setAsState();
  }

  public void extractNodes(){
    if(lNum!=9) {

     int l = subNodes.length();
      String w="";
      for(int i=0;i<l;i++){
        char ch = subNodes.charAt(i);
        if(ch!=',') {
          w+=ch;
        }else{

sub_nodes.add(comp.layers.get(lNum).searchById(w));
          w="";
        }
      }
    }
  }
}

public class Layer {
  ArrayList<Node> nodes = new ArrayList<Node>();
  int layerNum = 0;
```

```
  public Layer(int Num){
    layerNum = Num;
  }

  public void refreshLayer(){
    for(int i=0;i<nodes.size();i++){
      Node temp = nodes.get(i);
      if(temp.nodeType!=0)continue;
      temp.pref=0;
      for(int j=0;j<temp.sub_nodes.size();j++){
        temp.pref += (int)(temp.sub_nodes.get(j).pref)/2;
      }

//System.out.print(temp.pref!=0?temp.pref+"\n":"");
    }
  }

  public Node searchByState(int[][] state){
    String temp =
""+state[0][0]+state[0][1]+state[0][2]+state[1][0]+state[1][
1]+state[1][2]+state[2][0]+state[2][1]+state[2][2];
    //System.out.print(temp.!=0?temp:"");
    Node ret = searchById(temp);
    return ret;
  }
```

```
    public Node searchById(String id){
     Node ret = null;
     for(int i=0;i<nodes.size();i++){
       //System.out.println("*********"+nodes.get(i).id);
       if(nodes.get(i).id.equals(id)) {
         ret = nodes.get(i);
         break;
       }
     }
     return ret;
   }
}
```

Kode Lampiran 3 Kode TicTacToe untuk Pengujian

LAMPIRAN B : REKAPITULASI KUESIONER

Rekapitulasi Kuesioner berisi data keseluruhan yang didappatkan ketika kuesioner dilakukan. Tabel ini berisi jumlah total responden memilih untuk seetiap pilihan jawaban. Kuesioner diberikan kepada 20 orang narasumber yang berpengalaman di bidang pemrograman dan familiar dengan bahasa pemrograman Java. Berikut adalah rekapitulasi nilai dari kuesioner :

Tabel Lampiran 1 Rekapitulasi Kuisioner

Pertanyaan	Jumlah Responden yang Memilih
1. Sejauh mana aplikasi ini membantu Anda dalam proses pengukuran kualitas perangkat lunak?	
a. Sangat Membantu	6
b. Membantu	14
c. Cukup Membantu	-
d. Kurang Membantu	-
e. Tidak Membantu	-
2. Bagaimana pendapat Anda mengenai tampilan awal pada aplikasi ini?	
a. Sangat Baik	2
b. Baik	17
c. Cukup Baik	1
d. Buruk	-
e. Sangat Buruk	-

Pertanyaan	Jumlah Responden yang Memilih
3. Sejauh mana Anda memahami tampilan akhir atau laporan hasil perhitungan pada aplikasi ini?	
a. Sangat Memahami	10
b. Memahami	8
c. Cukup Memahami	2
d. Kurang Memahami	-
e. Tidak Memahami	-
4. Bagaimana pendapat Anda mengenai waktu yang dibutuhkan aplikasi ketika dijalankan?	
a. Sangat Cepat	20
b. Cepat	-
c. Cukup	-
d. Lama	-
e. Sangat Lama	-
5. Apakah aplikasi ini cukup mudah untuk digunakan?	
a. Sangat Mudah	13
b. Mudah	7
c. Cukup Mudah	-
d. Sulit	-
e. Sangat Sulit	-
6. Apakah warna, ukuran dan pemilhan jenis	

Pertanyaan	Jumlah Responden yang Memilih
tulisan pada tampilan sesuai?	
a. Sangat Sesuai	-
b. Sesuai	20
c. Cukup Sesuai	-
d. Kurang Sesuai	-
e. Tidak Sesuai	-
7. Apakah warna, ukuran dan pemilhan jenis tulisan pada hasil sesuai?	
a. Sangat Sesuai	-
b. Sesuai	20
c. Cukup Sesuai	-
d. Kurang Sesuai	-
e. Tidak Sesuai	-
8. Apakah informasi yang diberikan jelas ketika terjadi eror?	
a. Sangat Jelas	17
b. Jelas	3
c. Cukup Jelas	-
d. Kurang Jelas	-
e. Sangat Tidak Jelas	-
9. Apakah ukuran, bentuk, dan fungsi tombol sudah sesuai?	
a. Sangat Sesuai	7

Pertanyaan	Jumlah Responden yang Memilih
b. Sesuai	13
c. Cukup Sudah	-
d. Kurang Sesuai	-
e. Tidak Sesuai	-
10. Bagaimana pendapat Anda tentang keseluruhan aplikasi ini?	
a. Sangat Baik	10
b. Baik	10
c. Cukup Baik	-
d. Buruk	-
e. Sangat Buruk	-

BIODATA PENULIS

Penulis, Wati Margaretha Marpaung, lahir di kota Balige pada tanggal 11 Juli 1993. Penulis adalah anak ketiga dari empat bersaudara dan dibesarkan di desa Paritohan, Tobasa, Sumatera Utara.

Penulis menempuh pendidikan formal di SD Negeri 173651 Pintupohan Meranti (1999-2005), SMP Budhi Dharma Balige (2005-2008), SMA Budi Mulia Pematangsiantar (2008-2011). Pada tahun 2011, pennulis memulai pendidikan S1 jurusan Teknik Informatika Fakultas Teknologi Informasi di Institut Teknologi Sepuluh Nopember Surabaya, Jawa Timur.

Di jurusan Teknik Informatika, penulis mengambil bidang minat Rekayasa Perangkat Lunak dan memiliki ketertarikan di bidang SQA, basis data, *software evolution*, dan *software maintenance*. Penulis juga aktif dalam organisasi kemahasiswaan seperti Himpunan Mahasiswa Teknik Computer (HMTC). Dan penulis juga pernah menjadi asisten dosen mata kuliah Matematika Diskrit. Penulis dapat dihubungi melalui alamat email creatureofwati@gmail.com .